Transforming XML trees for efficient classification and clustering

Laurent Candillier[1,2], Isabelle Tellier[1], Fabien Torre[1]

[1] GRAppA - Charles de Gaulle University - Lille 3
http://www.grappa.univ-lille3.fr
candillier@grappa.univ-lille3.fr
[2] Pertinence - 32 rue des Jeûneurs -75002 Paris
http://www.pertinence.com

Abstract. Most of the existing methods we know to tackle datasets of XML documents directly work on the trees representing these XML documents. We investigate in this paper the use of a different kind of representation for the manipulation of XML documents.
Our idea is to transform the trees into sets of attribute-values, so as to be able to apply various existing methods of classification and clustering on such data, and benefit from their strengths. We apply this strategy both for the classification task and for the clustering task using the structural description of XML documents alone.
For instance, we show that the use of *boosted C5* [1] leads to very good results in the classification task of XML documents transformed in this way. The use of *SSC* [2] in the clustering task benefits from its ability to provide as output an interpretable representation of the clusters found. Finally, we also propose an adaptation of *SSC* for the classification of XML documents, so that the produced classifier is understandable.

1 Introduction

Most of the existing methods we know to tackle datasets of XML documents directly work on the trees representing these XML documents. Some methods are based on the use of metric distances to compare the trees: the *edit distance*, minimum number of mutations required to change a tree into another one [3, 4], or the number of paths they share [5–7]. Other methods aim at discovering frequent subtrees in the data [8, 9].

We investigate in this paper the use of a different kind of representation for the manipulation of XML documents. We propose to transform the trees into sets of attribute-values. In [10], the authors used such an idea to take into account the structure when classifying XML documents. But the representation they chose for the trees was a simple *bag-of-tags*. We propose to use richer features: the set of parent-child relations, the set of "next-sibling" relations, the set of paths starting from the root and the arity of the nodes.

The use of such representations allows us to apply various existing methods for classifying or clustering sets of XML documents. And we can thus benefit from the strengths of these methods. In particular, our experiments exhibit very good

results when applying *boosted C5* [1] on sets of XML documents transformed in this way. The use of *SSC* [2] for clustering tasks on such transformed datasets also benefits from its ability to provide as output an interpretable representation of the clusters found. Moreover, we also propose in this paper an adaptation of *SSC* for the classification of XML documents, so that the produced classifier is understandable.

The remainder of the paper is organized as follows: in section 2, we present our proposed methodology to transform a tree into a set of attribute-values, and how such new dataset can be used for classification tasks; in section 3, we describe the adaptation of the *SSC* algorithm for XML datasets; the results of our experiments, conducted on the datasets provided by the XML mining challenge at INEX [11] on the *structure only tasks*, are then reported in section 4; finally, section 5 concludes the paper and suggests topics for future research.

2 Tree transformation

There are many possible ways to transform a tree into a set of attribute-values. The first basic possibility is to transform the tree into the set of labels present at its nodes. An XML document would thus be transformed into a simple *bag-of-tags*. If the different sets of documents to be identified use different sets of tags, this representation would be sufficient to distinguish them. However, by doing that, we do not take into account the structure of the XML trees. To go further, we propose to construct the following attributes from a set of available trees:

- the set of parent-child relations (whose domain is the set of pairs of tags labelling the nodes);
- the set of next-sibling relations (whose domain is the set of pairs of tags labelling the nodes);
- the set of distinct paths (including sub-paths), starting from the root (whose domain is the set of finite sequences of tags labelling the nodes).

So, we create as many new attributes as distinct features are encountered in the training set. For each of them, their value for a given document is the number of their occurences in this document. Finally, we also define as many new attributes as there are absolute distinct node positions represented in the trees. The identifier of such a node position can be coded, for example, by a sequence of integers: the root is coded 0, its first child is coded 0.0, its second child 0.1, etc. For every identifier of a node position, the value of the attribute for a document is the arity of the node, that is the number of its child nodes in the document (whose domain is an integer). So the new introduced attributes all take their value into the set of natural numbers. As an intuition, such representation should allow to distinguish, for example:

- two sets of documents that use different tags, or in which the number of some given tags are different;

– one set of documents in which a given relation (parent-child or next-sibling) between some given tags is allowed, from another set that does not allow such a relation;
– or a set of documents in which the number of children of a given node position is different from the one in another set of documents.

Such representation could lead to a high number of generated attributes. So the algorithms used to tackle such new datasets should be able to handle many attributes, and to perform *feature selection* during their learning process. In a classification task, *C5* [1] is for example well suited. In a clustering task, a *subspace clustering* algorithm, that is a clustering algorithm able to characterize every distinct cluster on a limited number of dimensions (eventually distinct for each cluster) could be useful. We describe such a method in the next section.

3 Algorithm SSC

SSC [2] is a *subspace clustering* algorithm based on the use of a probabilistic model under the assumption that the clusters follow independent distributions on each dimension. It uses the well-known *EM algorithm* [12]. *SSC* has been shown to be effective, and it is able to provide as output an interpretable representation of the clusters found, as a set of rules, and a way to visualize them effectively. Moreover, a new step of *hard feature selection* has been added to keep only the best attributes for each cluster, and thus be less sensitive to irrelevant dimensions, and faster.

In the next subsections, we first formalize the different steps of the method. We then present an adaptation for facing datasets of XML documents, and another adaptation for supervised classification. One of the interests of this method is that it provides an output which can be represented by a hierarchy of tests.

3.1 Formalization of the method

Let us first introduce some notations. We denote by N the number of data points $\overrightarrow{X_i}$ of the input dataset D, and by M the number of dimensions on which they are defined. We only present here the case where the dimensions are numerical, but the adaptation for datasets containing also categorical dimensions can be found in [2].

The basis of our model is the classical mixture of probability distributions $\theta = (\theta_1, ..., \theta_K)$ where each θ_k is the vector of parameters associated with the k^{th} cluster to be found, denoted by C_k (we set to K the total number of clusters). In our model, we suppose that the data follow gaussian distributions. So the model has the following parameters θ_k for each cluster C_k: π_k denotes its weight, μ_{kd} its mean and σ_{kd} its standard deviation on dimension d. We then use the *EM algorithm* [12] to find the model parameters that best fit the data.

The E-step consists in computing the membership probability of each data point $\overrightarrow{X_i}$ to each cluster C_k with parameters θ_k. In our case, dimensions are

supposed to be independent. So the membership probability of a data point to a cluster is the product of membership probabilities on each dimension:

$$P(\overrightarrow{X_i}|\theta_k) = \prod_{d=1}^{M} \frac{1}{\sqrt{2\pi}\sigma_{kd}} e^{-\frac{1}{2}\left(\frac{X_{id}-\mu_{kd}}{\sigma_{kd}}\right)^2}$$

$$P(\overrightarrow{X_i}|\theta) = \sum_{k=1}^{K} \pi_k \times P(\overrightarrow{X_i}|\theta_k) \text{ and } P(\theta_k|\overrightarrow{X_i}) = \frac{\pi_k \times P(\overrightarrow{X_i}|\theta_k)}{P(\overrightarrow{X_i}|\theta)}$$

Then the M-step consists in updating the model parameters according to the new class probabilities as follows:

$$\pi_k = \frac{1}{N} \sum_i P(\theta_k|\overrightarrow{X_i})$$

$$\mu_{kd} = \frac{\sum_i X_{id} \times P(\theta_k|\overrightarrow{X_i})}{\sum_i P(\theta_k|\overrightarrow{X_i})} \text{ and } \sigma_{kd} = \sqrt{\frac{\sum_i P(\theta_k|\overrightarrow{X_i}) \times (X_{id}-\mu_{kd})^2}{\sum_i P(\theta_k|\overrightarrow{X_i})}}$$

These two steps iterate until a stopping criterion is reached. Usually, it stops when the *log-likelihood* of the model to the data, $LL(\theta|D) = \sum_i \log P(\overrightarrow{X_i}|\theta)$, increases less than a small positive constant δ from one iteration to another. But in order to cope with the problem of slow convergence with the classical EM algorithm, it has been shown in [2] that adding the following *k-means like* stopping criterion is effective: stop whenever the membership of each data point to their most probable cluster does not change from one iteration to another. To do this, we introduce a new view on each cluster C_k, corresponding to the set of data points belonging to it:

$$S_k = \{\overrightarrow{X_i}|Argmax_{j=1}^{K} P(\overrightarrow{X_i}|\theta_j) = k\}$$

The set of all S_k thus define a partition on the dataset.

And finally, to cope with the problem of sensitivity to the choice of the initial solution, we run the algorithm many times with random initial solutions and keep the model that optimizes the *log-likelihood* of the model to the data $LL(\theta|D)$.

Moreover, a new step of *hard feature selection* has been added to keep only the best attributes for each cluster. This is done by using a user parameter, denoted by *nb_ds*, that specifies how many attributes to keep for each cluster. Thus, for each cluster, the attributes of highest weights are kept, and the others are ignored. These weights W_{kd} are computed as the ratio between local and global standard deviations:

$$W_{kd} = 1 - \frac{\sigma_{kd}^2}{\sigma_d^2} \text{ , with } \sigma_d^2 = \frac{1}{N} \sum_i (X_{id} - \mu_{kd})^2$$

To make the results as comprehensible as possible, we now introduce a third view on each cluster, corresponding to its description as a rule defined with as few dimensions as possible.

Although we have already selected a subset of dimensions relevant for each cluster, it is still possible to prune some and simplify the clusters representation while keeping the same partition of the data. See figure 1 as an example. In this case, the cluster on the right is dense on both dimensions X and Y. So its true description subspace is $X \times Y$. However, we do not need to consider Y to distinguish it from the other clusters: defining it by high values on X is sufficient. The same reasoning holds for the cluster on the top and the dimension Y.

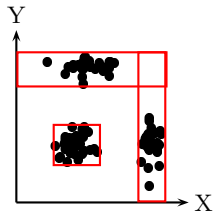

Fig. 1. Example of minimal description.

To do this *dimension pruning*, we first create the rule R_k associated with the current cluster C_k. We now only consider the set of dimensions considered as relevant according to the previous selection, and associate with the rule R_k the smallest interval containing all the coordinates of the data points belonging to S_k. We then compute the support of the rule (the set of data points comprised in the rule). This step is necessary since it is possible that some data points belong to the rule but not to the cluster. And then, for all relevant dimensions d presented in ascending order of their weights W_{kd}, we delete the dimension from the rule if the deletion does not modify its support.

3.2 Adaptation for clustering XML documents

For ease of interpretation, and to speed up the algorithm for clustering XML documents based on their structure, we propose to adapt *SSC* so that the dataset is recursively cut into two parts according to a given set of attributes associated with the trees. The output of our clustering method is then a hierarchy in which each node represents a test on various attributes.

Before giving more details on our procedure, let us introduce some notations. A denotes the set of possible attributes associated with the XML documents. As presented in the previous section, A can be partitioned into groups of attributes of increasing complexity:

- we call A_1 the set of tags,
- A_2 is the set of parent-child relations,
- A_3 is the set of next-sibling relations,
- A_4 is the set of node positions,
- and A_5 is the set of paths starting from the root.

A is thus composed of $SA = 5$ classes of attributes. Finally, we denote by $Cut(C_k, A_i)$ the partitionning into two parts of the dataset that is made of the documents included in the cluster C_k, and transformed with the given set of attributes A_i.

The main steps of our new clustering method are presented by algorithm 1. It consists in choosing at each step the cut of highest interest, among all the possible cuts of the current clusters C_k for $k \in [1..K]$ on the possible sets of attributes $A_i \in A$, until the desired number of clusters is reached. The interest of partitionning the elements of the cluster C_k with the attributes of A_i is computed as the ratio between the *log-likelihood* of a partition with two clusters and the *log-likelihood* of a partition with only one cluster, weighted by the number of data points in C_k, to prefer the divisions of clusters containing many data points. The output of the procedure is a hierarchy in which each node corresponds to a membership test to a rule, created as presented in the previous subsection, and defined with as few attributes as possible.

Algorithm 1 SSC for XML

Input: the dataset D of XML documents and the desired number of clusters nb_clus
- set K, the current number of clusters of the partition, to 1
- initialize the unique cluster C_1 with all the documents of the dataset D
- create a new empty hierarchy H

while $K \neq nb\_clus$ **do**
  for all $k \in [1..K]$ **do**
    for all $i \in [1..SA]$ **do**
      - compute the interest of $Cut(C_k, A_i)$
    end for
  end for
  - select and perform the cut of highest interest
  - compute the associated rule and add it to the hierarchy H
end while

Output: the hierarchy H, with $nb\_clus - 1$ tests

3.3 Adaptation for understandable classification

In order to benefit from the ability of the previously presented method to provide understandable results, we now propose to adapt it for supervised classification. The new method has two main steps: a first step that is *clustering like* but uses the classes, and a second step that is completely guided by the classes.

The first step is described by algorithm 2 and consists in a clustering phase that allows to mix various classes in one cluster but does not allow a class to be splitted into different clusters. At this step, we prefer that a cutting rule is defined on the tags attributes better than on any other set of attributes, because they are *simpler* attributes. In the same way, we prefer to use attributes representing relations between tags (parent-child or next-sibling) than paths information. So

more generally, we prefer using A_i than A_j if $i < j$. That's why we perform a cut each time it is possible, rather than comparing the interest on various possible cuts, as is done for clustering. The same reasoning will also holds in the next step.

Algorithm 2 Step 1

Input: the dataset D of XML documents
- initialize the unique cluster C_1 with all the documents of the dataset D
- create a new empty hierarchy H
- set $CUT = 1$

while $CUT = 1$ **do**
- set $CUT = 0$

for all $k \in [1..K]$ **do**
- set $CUT_k = 0$ and $i = 1$

while $CUT_k = 0$ and $i <= SA$ **do**

if in $Cut(C_k, A_i)$, no class is splitted into different parts **then**
- perform the partitionning
- compute the associated rules and update the hierarchy
- set $CUT_k = 1$ and $CUT = 1$

else
- i = i + 1

end if

end while

end for

end while

Output: the hierarchy H, and the current partition

The second step of our method is described by algorithm 3. It takes as input the output of the previous step and consists in separating the classes that are still embedded in the same clusters. It is itself composed of two main steps: the first one tries to distinguish the classes using rules, in order to be as understandable as possible, while the second one uses probabilistic models that are richer models, able to fit more complex decision surfaces.

1. If a rule found is able to discriminate one class from the others, then this rule is used as the next test in the hierarchy. As has been motivated earlier, a split is performed as soon as possible.
2. Then, if no rule has been found that is able to discriminate one class from the others in one given cluster, we test the error rates obtained in cross-validation with probabilistic models generated on each possible sets of attributes, and select the one that leads to the lowest error rate as the next test in the hierarchy.

So in the final hierarchy, the tests on the nodes can be of two different natures: they can correspond to membership tests to rules, or to probability tests on probabilistic models. Each of these tests are perfomed on only one set of attributes at a time.

Algorithm 3 Step 2

Input: the hierarchy H and the partition from the first step
  for all $k \in [1..K]$ **do**
    while C_k contains different classes **do**
      for all $i \in [1..SA]$ **do**
        for all $class \in C_k$ **do**
          if a rule is able to distinguish the class from the others **then**
            - perform the partitionning and update the hierarchy
          end if
        end for
      end for
      if no split has been done **then**
        for all $i \in [1..SA]$ **do**
          - compute the classification error rate in cross-validation of the probabilistic model generated on the attributes A_i
        end for
        - choose the model that leads to the lowest error rate and update the hierarchy
      end if
    end while
  end for
Output: the hierarchy H

4 Experiments

Our experiments were conducted on the datasets provided by the XML mining challenge at INEX on the *structure only tasks*. The classification method presented in section 2 was applied on all datasets: *inex-s*, *m-db-s-0*, *m-db-s-1*, *m-db-s-2* and *m-db-s-3*. The results are presented in the first subsection. The other two methods presented in section 3 were only applied on the *m-db-s-0* dataset, due to the lack of time. Their results are presented in the second subsection.

4.1 Boosted C5 on the transformed datasets

Table 1 presents the number of new attributes generated by transforming the XML trees into attribute-values for each dataset. We can thus observe that our method creates many new attributes. In particular, the number of attributes representing the paths in the trees is very high.

We applied the algorithm C5 [1] boosted 10 times on these datasets. However, as the number of attributes was too high for C5 on the *m-db-s-2* and *m-db-s-3* datasets, we did not use the attributes representing the paths in the trees for these datasets. The error rates obtained in 10-fold cross-validations on all the datasets are provided in table 2, and show that our proposed methodology has reasonable error rates.

Table 3 then reports the *micro-recall* and *macro-recall* computed on the test datasets using this method. Datasets *m-db-s* are all based on the same initial dataset but each succesive dataset contains more noise than the previous one. The results of our method thus shows that it is robust to the presence of noise.

dataset	number of tags	number of parent-child relations	number of next-sibling relations	number of node positions	number of paths	total
inex-s	150	1038	827	2475	3674	8164
m-db-s-0	197	2172	419	6575	320	9683
m-db-s-1	197	6477	5617	9159	16772	38222
m-db-s-2	196	8953	7455	9183	25628	51415
m-db-s-3	199	10639	9557	8537	37576	66508

Table 1. Number of attributes generated for each dataset.

dataset	error rate
inex-s	0.011
m-db-s-0	0.026
m-db-s-1	0.038
m-db-s-2	0.062
m-db-s-3	0.062

Table 2. Error rates of boosted C5 on the datasets transformed into attribute-values.

dataset	micro-recall	macro-recall
inex-s	0.941	0.958
m-db-s-0	0.968	0.960
m-db-s-1	0.966	0.956
m-db-s-2	0.942	0.932
m-db-s-3	0.947	0.935

Table 3. Micro-recall and macro-recall of boosted C5 on the test datasets transformed into attribute-values.

4.2 Adaptations of SSC

For reasons of time, the algorithms adapted from *SSC* were only experimented on the *m-db-s-0* dataset. The parameter *nb_ds* defined in section 3, and representing the number of most relevant dimensions to be selected for each cluster, was set to 10.

Our clustering method managed to identify correctly the classes number 1 to 5. It mixed classes number 7 and 9 together, and the remaining classes were also mixed together. This clustering thus leads to a *micro purity* of 0.78, a *macro purity* of 0.75, a *micro entropy* of 0.18, a *macro entropy* of 0.21, and a *mutual information* of 1.87 on test data.

The hierarchy formed is presented by figure 2. $R4$, $R10$, $R6$, and $R7$ are conjunctions of 10 tests:

- $R4$ is defined on the attributes representing the next-sibling relations,
- $R10$ concerns the number of children according to the node positions,
- $R6$ is defined on the attributes representing the tags,
- and $R7$ concerns the number of children according to the node positions.

Finally, $R11$ tests whether the number of paths (BE-AL-AT) is lower or equal to 1, and if there is no path (BE-AL-AT-AR).

The adaptation of our method for supervised learning leads to very interesting results. The hierarchy obtained is presented by figure 3. We can thus observe that the given hierarchy is very understandable.

- $S2$, $S3$, $S4$ and $S5$ represent the probabilities on models based on next-sibling relations, respectively concerning classes 2, 3, 4 and 5.
- $P6$ and $P11$ represent the probabilities on models based on the paths of the trees, concerning classes 6 and 11.
- And $Nb(0.0.0)$ indicates the number of children of the first grand-child of the root.

Thus, for example, the membership to class 1 only depends on the presence of the tag named *movie*. And in the same way, the membership to class 8 only depends on the absence of the tags *movie*, CL, BJ, AJ, and the presence of the parent-child relation between tags AT and AQ. The error rate of this tree on the train dataset is 0.03. It misclassified very few documents, except those of classes 6 and 11 that were mixed.

The results of this method on test data were also very reasonable, leading to a *micro-recall* of 0.906 and a *micro-recall* of 0.924.

5 Conclusion

We have shown in this paper that transforming XML document trees into sets of attribute-values can lead to very promising results, provided that these attributes are considered as sets of increasing complexity. This representation allows us to

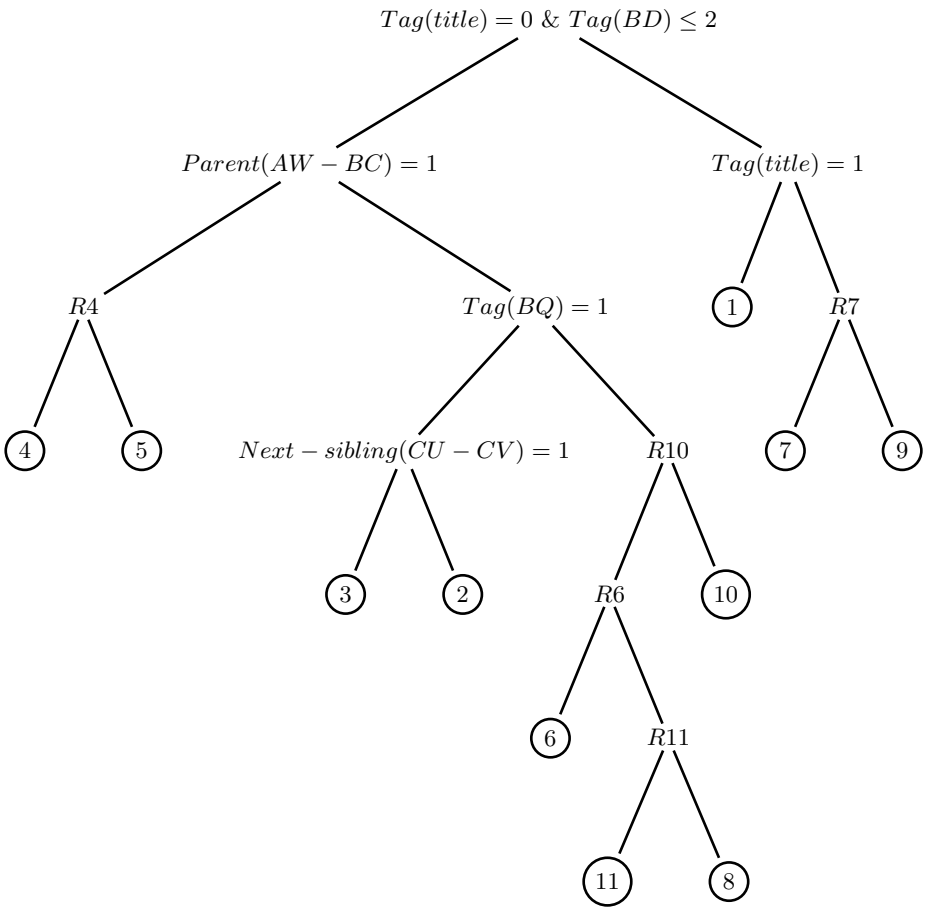

Fig. 2. Tree obtained when clustering dataset m-db-s-0.

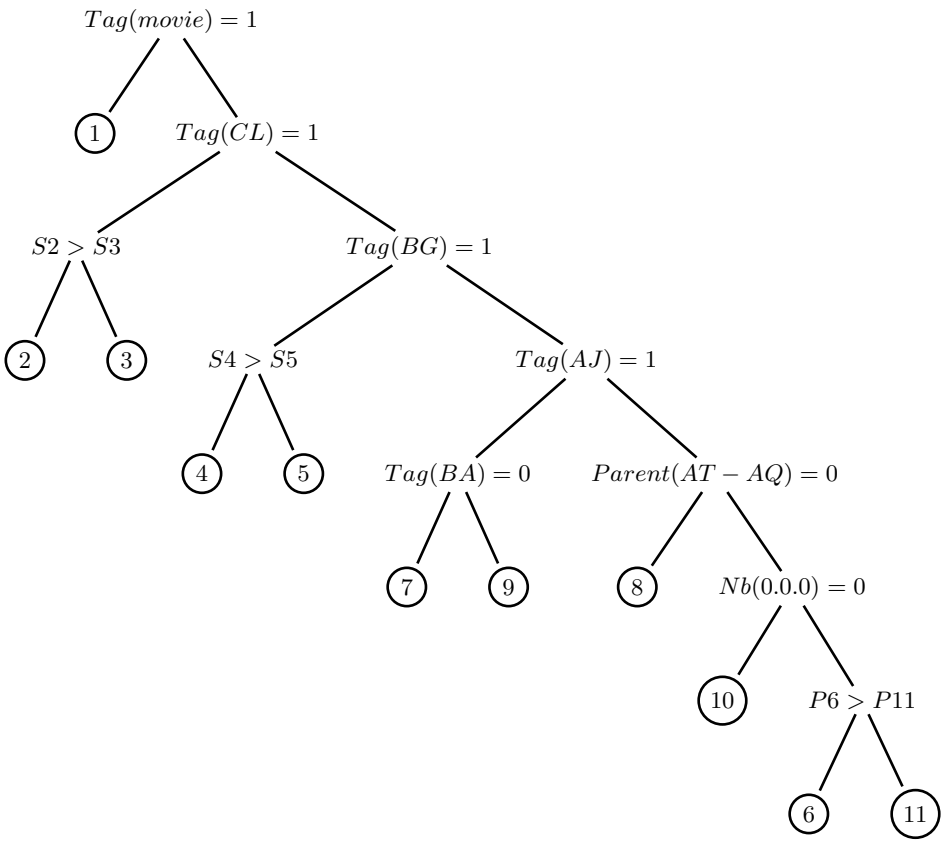

Fig. 3. Tree obtained for classifying dataset m-db-s-0.

benefit from the strengths of existing methods. We have also presented a new original method able to provide an interpretable classifier as an output.

We conjecture it is now possible to go further in the way we transform the trees. For instance, it is possible to consider as new attributes some forks of the trees, of some given height and width, or to identify in which part of the trees the tags or relations between tags are present. But, as has been shown in the experiments part of the paper, we already constructed a lot of new attributes with our method. And by using such attributes, we already obtain very good results.

To take into account such possible differences between trees, a special care should now be taken to find a compromise between the number of new created attributes and the information they carry. This point should be studied in further researchs.

References

1. Quinlan, R.: Data mining tools see5 and c5.0 (2004)
2. Candillier, L., Tellier, I., Torre, F., Bousquet, O.: SSC : Statistical Subspace Clustering. In Perner, P., Imiya, A., eds.: 4th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM'2005). Volume LNAI 3587 of LNCS., Leipzig, Germany, Springer Verlag (2005) 100–109
3. Nierman, A., Jagadish, H.V.: Evaluating structural similarity in XML documents. In: 5th International Workshop on the Web and Databases (WebDB 2002), Madison, Wisconsin, USA (2002)
4. Dalamagas, T., Cheng, T., Winkel, K.J., Sellis, T.: Clustering XML documents by structure. In: 3rd Hellenic Conference on Artificial Intelligence, Samos, Greece (May 2004)
5. Flesca, S., Manco, G., Masciari, E., Pontieri, L., Pugliese, A.: Detecting structural similarities between XML documents. In: 5th International Workshop on The Web and Databases (WebDB'02), Madison, Wisconsin (2002)
6. Lian, W., Cheung, D.W., Mamoulis, N., Yiu, S.M.: An efficient and scalable algorithm for clustering XML documents by structure. IEEE transactions on Knowledge and Data Engineering **16** (January 2004) 82–96
7. Costa, G., Manco, G., Ortale, R., Tagarelli, A.: A tree-based approach to clustering XML documents by structure. Technical report, Institute of Italian National Research Council, Rende, Italy (April 2004)
8. Termier, A., Rousset, M.C., Sebag, M.: Treefinder: a first step towards xml data mining. In: IEEE International Conference on Data Mining (ICDM02). (2002) 450–457
9. Zaki, M.J., Aggarwal, C.C.: Xrules: An effective structural classifier for xml data. In: SIGKDD 03, Washington, DC (2003)
10. Doucet, A., Ahonen-Myka, H.: Naïve clustering of a large XML document collection. In: 1st Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX'02), Schloss Dagstuhl, Germany (2002)
11. Denoyer, L., Gallinari, P., Vercoustre, A.M.: XML Mining Challenge at INEX 2005. Technical report, University of Paris VI, INRIA (2006)
12. Ye, L., Spetsakis, M.: Clustering on unobserved data using mixture of gaussians. Technical report, York University, Toronto, Canada (Oct. 2003)