International Journal of Pattern Recognition and Artificial Intelligence

Hybrid Decision Tree Architecture utilizing Local SVMs for Efficient Multi-Label Learning

Dejan Gjorgjevikj

Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University
Rudgjer Boshkovikj 16, 1000 Skopje, Macedonia
dejan.gjorgevikj@finki.ukim.mk

Gjorgji Madjarov

Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University
Rudgjer Boshkovikj 16, 1000 Skopje, Macedonia
gjorgji.madjarov@finki.ukim.mk

Sašo Džeroski

Department of Knowledge Technologies, Jožef Stefan Institute
Jamova cesta 39, 1000 Ljubljana, Slovenia
saso.dzeroski@ijs.si

Multi-label learning (MLL) problems abound in many areas, including text categorization, protein function classification, and semantic annotation of multimedia. An issues that severely limits the applicability of many current machine learning approaches to MLL are the large-scale problem, which have a strong impact on the computational complexity of learning. These problems are especially pronounced for approaches that transform MLL problems into a set of binary classification problems for which SVMs are used. On the other hand, the most efficient approaches to MLL, based on decision trees, have clearly lower predictive performance. We propose a hybrid decision tree architecture, where the leaves do not give multi-label predictions directly, but rather utilize local SVM-based classifiers giving multi-label predictions. A binary relevance architecture is employed in the leaves, where a binary SVM classifier is built for each of the labels relevant to that particular leaf. We use a broad range of multi-label datasets with a variety of evaluation measures to evaluate the proposed method against related and state-of-the-art methods, both in terms of predictive performance and time complexity. Our hybrid architecture on almost every large classification problem outperforms the competing approaches in terms of the predictive performance, while its computational efficiency is significantly improved as a result of the integrated decision tree.

Keywords: multi-label; learning; hybrid; architecture; decision tree; SVM.

1. Introduction

Single-label classification is concerned with learning from examples, each associated with a single label λ_i from a finite set of disjoint labels $\mathcal{L} = \{\lambda_1, \lambda_2, ..., \lambda_Q\}$, $|\mathcal{L}| > 1$. The task is to learn a mapping l: $\mathcal{X} \rightarrow \mathcal{L}$ (where $\mathcal{X}$ denotes the example space) that assigns a single label to each example. For $|\mathcal{L}| > 2$, the task is referred to as a

multi-class classification.

In multi-label classification (MLC), the task is to learn a mapping h: $\mathcal{X} \rightarrow 2^{\mathcal{L}}$. Each example $x \in \mathcal{X}$ is associated to a set of labels $\mathcal{Y} \subseteq \mathcal{L}$. In contrast to multi-class classification, the labels are not assumed to be mutually exclusive, i.e., an example can be a member of more than one class. The labels in $\mathcal{Y}$ are called relevant and those in $\mathcal{L} \backslash \mathcal{Y}$ irrelevant for a given example.

Label ranking studies the problem of learning a mapping from a set of examples to rankings over a finite number of predefined labels. It can be considered a natural generalization of conventional (multi-class) classification, where instead of requesting only a single label (a top label), a ranking of all the labels is performed.

Besides the concept of multi-label classification, the multi-label learning concept includes the concept of *multi-label ranking* [1], which is understood as learning a model that associates the query example x both with a label ranking of the complete label set $\{\lambda_1, \lambda_2, ..., \lambda_Q\}$ and a bipartite partition of this set into relevant and irrelevant labels.

The issue of learning from multi-label data has recently attracted significant attention from many researchers. They are motivated from an increasing number of new applications, such as semantic annotation of images and video (news clips, movies clips), functional genomics (gene and protein function), music categorization into emotions, text classification (news articles, web pages, patents, emails, bookmarks, ...), directed marketing and others.

In recent years, many different approaches have been developed to solve the multi-label learning problems. Tsoumakas and Katakis [2] summarize them into two main categories: a) algorithm adaptation methods, and b) problem transformation methods. Algorithm adaptation methods extend specific learning algorithms to handle multi-label data directly. Examples include lazy learning [3] [4] [5], neural networks [6] [7], boosting [8] [9], classification rules [10], decision trees [11] [12] etc. Problem transformation methods, on the other hand, transform the multi-label learning problem into one or more single-label classification problems. The single-label classification problems are solved with a commonly used single-label classification approach and the output is transformed back into a multi-label representation via some reverse process. A common approach for problem transformation is to use class binarization methods, i.e. decomposition of the problem into several binary sub-problems that can then be solved using a binary base classifier. The simplest strategies in the multi-label setting are the one-against-all and one-against-one strategies, also referred to as the binary relevance method (BR) [2] and pair-wise method [13] [14] respectively.

Two major classes of algorithms for multi-label learning are decision-tree-based and SVM-based approaches. Algorithms from the first group are extremely efficient, but not very accurate. Algorithms from the second group, represented by the problem transformation SVM architectures [2] are very accurate, but can be computationally expensive, especially in large-scale problems and when labels abound.

In this paper, we propose a novel architecture that combines the ML-C4.5 deci-

sion tree [11] and the BR architecture (that uses SVMs as base classifiers for solving the partial binary classification problems). In this way we achieve effective and computationally efficient multi-label learning on large-scale problems with a large number of labels. This approach will be referred to as SVM-based decision trees for multi-label learning (ML-SVMDT).

Section 2 surveys previous related work in multi-label learning and highlights the problems of high dimensionality of the label space and large size of the datasets. The architecture that we propose as an effective and computationally efficient solution to these two problems is presented in Section 3. Section 4 presents the experimental setup and performance evaluation measures, while the experimental results that compare the performance of our architecture to the other competing methods are presented in Section 5. The conclusions are given in Section 6, along with some directions for further work.

2. Related work

In this section, we will give an overview of different methods for solving multi-label learning problems. These methods can be summarized in three main categories: Algorithm adaptation methods, problem transformation methods and ensemble methods. Additionally, the problem transformation methods can be grouped in three subcategories: Binary relevance methods, label power-set methods and pair-wise methods.

2.1. *Algorithm adaptation methods*

ML-C4.5 is an adaptation of the well known C4.5 algorithm for multi-label learning [11]. Clare et al. modified the formula of entropy calculation (equation 1) in order to solve multi-label problems. They also allowed multiple labels in the leaves of the tree. The modified entropy sums the entropies for each individual class label.

$$\mathcal{E}(S) = -\sum_{i=1}^{N} (p(c_i) \log p(c_i) + q(c_i) \log q(c_i)) \tag{1}$$

where S is the set of examples, $p(c_i)$ is the relative frequency of class label c_i and $q(c_i) = 1 - p(c_i)$. Each leaf is represented by the most frequent set of class labels that are associated to the training examples that belong to that leaf. If more than 50% of the training examples in the leaf are labeled with a particular label then that label belongs to the set of most frequent labels. The key property of ML-C4.5 is its computational efficiency. It is among the fastest and most computationally efficient multi-label classifiers available today.

Zhang et al. [15] proposed an adaptation of Random Decision Tree (RDT) for multi-label classification. The main difference between RDT and other classical decision tree methods, such as ML-C4.5, is that RDT constructs the decision tree randomly without using any information for labels.

Predictive clustering trees (PCTs) [12] are decision trees viewed as a hierarchy of clusters: the top-node corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. PCTs are constructed using a standard top-down induction of decision trees algorithm, where the variance and the prototype function can be instantiated according to the task at hand. Namely, PCTs can handle several types of structured outputs: tuples of continuous or discrete variables, time series, classes organized into a hierarchy, tuples of time series and tuples of hierarchies [16]. For the task of predicting tuples of discrete variables, the variance function is computed as the sum of the *Gini indices* of the variables from the target tuple, i.e., $Var(E) = \sum_{i=1}^{T} Gini(E, Y_i)$. The prototype function returns a vector of probabilities that an example belongs to a given class for each variable from the target tuple. In the case of multi-label learning, it returns a vector of probabilities that an example is labelled with a given label.

ML-kNN [3] is based on the popular k Nearest Neighbours (kNN) lazy learning algorithm. The first step in this approach is the same as in kNN, i.e., retrieving the k nearest examples. It uses the maximum a posteriori principle in order to determine the label set of the test example, based on prior and posterior probabilities i.e. the frequency of each label within the k nearest neighbours. Other kNN based approaches for multi-label learning also exist [17 4 5].

Neural networks have also been adapted for multi-label classification [6 7]. BP-MLL [7] is an adaptation of the popular back-propagation algorithm for multi-label learning. The main modification to the algorithm is the introduction of a new error function that takes multiple labels into account.

2.2. *Problem transformation methods*

2.2.1. *Binary relevance methods*

An extensive bibliography of learning algorithms for problem transformation methods is given by Tsoumakas and Katakis [2]. The simplest strategy in the multi-label setting is the one-against-all strategy also referred to as the binary relevance method (BR) [2]. It addresses the multi-label learning problem by learning one classifier for each class, using all the examples labelled with that class as positive examples and all remaining examples as negative examples. At query time, each binary classifier predicts whether its class is relevant for the query example or not, resulting in a set of relevant labels. In the ranking scenario, the labels are ordered according to the probability association of each label from each binary classifier. The most important and widely relevant advantage of BR is its low computational complexity compared to other methods. It is theoretically simple and intuitive. Its assumption of label independence makes it suited to contexts where new examples may not necessarily be relevant to any known labels or where label relationships may change over the test data. Given a constant number of examples, BR scales linearly with the size of the known label set Q. A method closely related to the BR method is the Classifier

Chain method (CC) proposed by Read et al. [18]. This method involves Q binary classifiers as in BR. Classifiers are linked along a chain where each classifier deals with the binary relevance problem associated with label $\lambda_i \in L$, $(1 \leq i \leq Q)$. The feature space of each link in the chain is extended with the 0/1 label associations of all previous links. The ranking and the prediction of the relevant labels in the CC method are the same as in the BR method. Xu [19] propose one more BR method that extends the traditional binary support vector machine for multi-label classification.

2.2.2. *Label power-set methods*

The second problem transformation method is the label combination method, or label power-set method, (LP), which has been the focus of several recent studies [20] [21] [2]. The basis of this method is to combine entire label sets into atomic (single) labels to form a single-label problem for which the set of possible single labels represents all distinct label subsets in the original multi-label representation. Each $(x, \mathcal{Y})$ is transformed into (x, l) where l is the atomic label representing a distinct label subset. In this way, LP based methods directly take into account label correlations. To solve the problem of the large number of label combinations, Read [22] developed a pruned problem transformation method (PPT), that selects only the transformed labels that occur more than predefined number of times. Tai et al. [23] proposed a novel method, Principle Label Space Transformation, that captures the important correlations between labels using a flat (a low-dimensional linear subspace) in the high-dimensional space. The method only uses a simple linear encoding of the vertices and a simple linear decoding of the reduced predictions, both easily computed from the Singular Value Decomposition of a matrix composed of the label-set vertices.

2.2.3. *Pair-wise methods*

Third problem transformation approach to solving the multi-label learning problem by using binary classifiers is pair-wise classification or round robin classification [13] [14]. Its basic idea is to use $Q * (Q-1)/2$ classifiers covering all pairs of labels. Each classifier is trained using the samples annotated with the first label as positive examples and the samples annotated with the second as negative examples. The sample annotated with both labels cannot be viewed as either a positive or a negative example, and because of that it is not involved in the learning process of the corresponding model. To combine these classifiers, the pairwise classification method naturally adopts the majority voting algorithm. Given a test example, each classifier delivers a prediction for one of the two labels. This prediction is decoded into a vote for one of the labels. After the evaluation of all $Q * (Q-1)/2$ classifiers, the labels are ordered according to their sum of votes. To predict the relevant and irrelevant labels for each example a thresholding method is used. The labels that accumulated votes above the chosen threshold are considered relevant and the rest

of the label are considered irrelevant for the corresponding test example.

Brinker et al. [1] and Fürnkranz et al. [24] propose a conceptually new technique for extending the common pair-wise learning approach to the multi-label scenario named Calibrated Label Ranking (CLR). The key idea of calibrated label ranking is to introduce an artificial (calibration) label λ_0, which will represent the split-point between relevant and irrelevant labels. The calibration label λ_0 is assumed to be preferred over all irrelevant labels, but all relevant labels are preferred over it. At prediction time (when majority voting is usually used), one will get a ranking over $Q+1$ labels (the Q original labels plus the calibration label λ_0). CLR is considered a combination of multi-label classification and ranking.

Besides majority voting in CLR, Park et al. [25] propose another, more effective voting algorithm named Quick Weighted Voting (QWeighted) for multi-class classification. QWeighted computes the class with the highest accumulated voting mass, while avoiding the evaluation of all possible pairwise classifiers. An adaptation of QWeighted to multi-label learning (QWeightedML) [26] is to repeat the process while all relevant labels are not determined, i.e., until the returned label is the artificial label λ_0, which means that all remaining labels will be considered to be irrelevant. Madjarov et al. [27] propose a novel architecture for efficient pair-wise multi-label learning, named Two Stage Architecture. The authors achieved a significant reduction in the computational complexity int the prediction pahse by introducing a threshold that dynamically affects the number of consulted pair-wise classifiers.

2.3. *Ensemble methods*

Several ensemble approaches have been developed based on the common algorithm adaptation and problem transformation methods. AdaBoost.MH and AdaBoost.MR [8] are two extensions of AdaBoost for multi-label data. While AdaBoost.MH is designed to minimize Hamming loss, AdaBoost.MR is designed to find a hypothesis which places the correct labels at the top of the ranking. A combination of AdaBoost.MH with an algorithm for producing alternating decision trees [9] has been proposed, with the motivation of producing multi-label models that can be understood by humans.

Problem transformation ensembles are the RAKEL system by Tsoumakas et al. [20] and ensembles of classifier chains (ECC) [28]. For m iterations of the training data, RAKEL draws a random subset of size k from all labels L and trains a label power-set classifier using these labels. A simple voting process determines the final classification set. Note that binary methods are occasionally referred to as ensemble methods because they involve multiple binary models. Li et al. [29] introduce a novel multi-label classification framework called Variable Pairwise Constraint projection for Multi-label Ensemble (VPCME) to construct a multi-label ensemble for handling multi-label data. This framework involves two inherent components, i.e., the variable pairwise constraint projection and the boosting-like strategy. However, none of these models is multi-label itself and therefore we use the term ensemble

strictly in the sense of an ensemble of multi-label methods. Kong et al. [30] proposed method for multi-label stream classification based on an ensemble of fading random trees, that can efficiently process high-speed multi-label stream data with concept drifts.

2.4. *Other methods related to multi-label learning*

Multi-task learning [31] [32] [33] and structured output prediction learning are domains closely related to the concept of multi-label learning. Multi-task learning studies the problem of learning data representations that are common across multiple related supervised learning tasks. It exploit the relations between tasks (classes) to enhance the performance of learning algorithms by simultaneously learning classifiers for multiple tasks.

Structured output prediction is a machine learning and regression technique that involve predicting structured objects [34] [35] [36] [12] [16] [37]. For example, the problem of translating a natural language sentence into a semantic representation such as a parse tree can be seen as a structured prediction problem in which the structured output domain is the set of all possible parse trees. Structured prediction generalizes supervised learning where the output domain is usually a small or simple set. A tree based method for structured output prediction using a kernelization of the algorithm is proposed by Guerts et. al [37].

Freng et al. [38] formalize the framework for applying Error-correcting Codes on multi-label classification. They studied the use of four classic Error-correcting Codes designs: repetition code, Hamming code, BCH code and low-density parity-check code.

2.5. *The problem of large datasets and large label spaces*

Recently, one of the most challenging issues in multi-label learning is the high dimensionality of the label space in the large-scale problems. Usually, this issue includes the problems of imbalance of the examples per certain labels in the datasets and the small number of training examples annotated with a particular label. These problems can significantly influence the computational complexity and the predictive performance of the previously proposed methods for multi-label learning. Some methods achieve higher computational efficiency at the cost of predictive accuracy, such as ML-kNN [3], ML-C4.5 [11], etc. These methods usually belong to the group of algorithm adaptation methods. Other proposed methods, based on problem transformation, use base classifiers with lower time complexity, such as Naive Bayes [39] [21], the one-layer perceptron [26], etc., in order to reduce the computational complexity. Most of the proposed methods that show good predictive performances on the standard datasets (dataset that do not include the previously mentioned problems) become unusable for these kind of problems.

Nasierding et al. [40] presents a novel multi-label classification framework for domains with large number of labels. The proposed framework comprises an initial

clustering phase that breaks the original training set into several disjoint clusters of data. It then trains a multi-label classifier from the data of each cluster. Given a new test instance, the framework first finds the nearest cluster and then applies the corresponding model.

Another computationally efficient multi-label classification method specifically designed for large multi-label datasets is HOMER [21]. It constructs a hierarchy of multi-label classifiers, each one dealing with a much smaller unique set of labels compared to Q (the total number of labels) and a more balanced example distribution. This leads to improved predictive performance and also to linear training and logarithmic testing complexities with respect to Q. One of the main processes within HOMER is the even distribution of a set of labels into k disjoint subsets so that similar labels are placed together and dissimilar apart. The best predictive performance is reported using a balanced k means algorithm customized for HOMER.

2.6. *Combining decision trees and SVMs*

Several approaches that combine decision trees and SVMs have been proposed for binary and multi-class classification. For example, Kumar et al. [41] propose a method that combines decision trees and global SVM models (models learned on the whole dataset) for solving binary classification problems. Other approaches, such as [42] [43], use the structure of the decision tree to organize/arrange SVM classifiers in its nodes in order to improve the computational efficiency and the predictive performance. Boosting ensemble of support vector machines for multi-class classification was proposed in [44]. Gama [45] proposed functional trees for multi-class classification and regression problems. In each node of the functional tree a classifier or regressor is learned depending on the type of the problem. The feature vectors of the examples are extended with the predictions of the classifier or the regressor and the splitting of the examples in the corresponding node can also be done by one of the "new" attributes. However, none of these approaches deal with MLC problems.

3. Integration of Decision Trees and local SVMs

In this paper, we propose a novel hybrid approach that integrates Decision Trees and Support Vector Machines (ML-SVMDT) for computationally efficient multi-label learning in domains with a large number of labels. This approach combines the algorithm adaptation method ML-C4.5 [11] and the problem transformation method Binary Relevance (BR) [2]: The latter uses SVMs as base classifiers for solving the partial binary classification problems. The main idea of the proposed approach is to exploit the advantages of both methods - the low computational complexity of ML-C4.5 and the predictive accuracy of the BR architecture with SVM classifiers.

3.1. *Local models and ML-SVMDT*

An approach to achieve effective and computationally efficient multi-label learning is to partition the global classification problem first and then learn local classifiers for each of those partitions (subproblems) separately. In the prediction phase, one first tries to determine the partition to which a test example belongs, and then to classify the example using a local classifier trained using the examples belonging to that partition only. The logic behind this approach is that the examples that belong to the same partition as the test example, would be able to provide more accurate and faster prediction compared to the global classifier. We propose a novel hybrid architecture that introduces local models based on support vector machine classifiers for solving multi-label learning problems.

Our approach combines the ML-C4.5 method for partitioning the input feature space and the BR method for local classification. The BR method uses SVMs as base classifiers for solving the partial binary classification problems. The main idea is to use the advantages of both methods in order to build an architecture that will improve the predictive performance and the computational efficiency of its constituents. This approach will be referred to as hybrid decision tree architecture utilizing local SVMs for efficient multi-label learning (ML-SVMDT).

Throughout the literature, BR is often mentioned, but consistently criticized on account of its assumption of label independence. Namely, during the transformation process, BR ignores label correlations that exist in the training data. BR predicted label sets are likely to contain either too many or too few labels, or labels that would never co-occur in practice. In order to reduce these inconsistencies in the multi-label prediction of the BR method, we employed the ML-C4.5 method. In this context, the ML-C4.5 method actually tries to separate the training examples into groups where the label inconsistency is eliminated. It finds the correlations between labels and splits the global problem into several local subproblems accordingly.

The implementation of the architecture consists of three separate, sequential phases.

(1) In the first phase, the decision tree is built according to the ML-C4.5 approach (Figure 1).
(2) In the second phase, the decision tree is pruned (Figure 3).
(3) In the third phase, local SVM models are learned first; the architecture is then built by integrating the decision tree and the local SVM models (Figure 4).

The procedure of building the hybrid decision tree utilizing local SVMs is outlined on Figure 1.

We first introduce the notation used on Figures 1, 2, 3 and 4 and then analyse each implementation phase separately. S^{train} represents the training dataset, while S^{val} is the validation dataset used for pruning. f represents one dataset feature, while g is the normalized information gain, calculated as the difference in entropy between a given dataset and its splits made according to f. $\mathcal{P}$ is a partition induced

by f on a particular dataset. g^* is the largest information gain achieved by splitting a given dataset according to the feature f^*, while $\mathcal{P}^*$ is the corresponding partition induced by f^*.

The procedure of building the decision tree according to the ML-C4.5 approach (Figure 2) takes a set of examples (S^{train}) as input and outputs a tree. The building process starts at the root of the tree with choosing one feature (f) of the data that most effectively splits the dataset of examples into subsets ($\mathcal{P}$). The criterion is the normalized information gain (difference in entropy g) that results from choosing a feature for splitting the data (line 6 of BestFeature procedure on Figure 1). The feature with the highest normalized information gain is chosen to make the decision. The process continues recursively in each node until one of the stopping criteria is reached. The function Acceptable($\mathcal{P}$) determines if splitting the data in a node using the feature f would violate any of the stopping criteria. In our approach we use three separate stopping criteria: all examples in a node are labelled with the same labels; no partitioning provides any information gain and the number of samples in any of the child nodes after the split would be below some predetermined minimal number of examples. The data in a node can be acceptably splitted if none of the stopping criteria is satisfied. If no suitable feature can be found that can still acceptably split the data, the process of splitting the dataset stops and the corresponding node is declared as a leaf. Each leaf represents a set of labels that is assigned to an example that reaches the leaf in the recognition phase. The set of labels for each leaf is defined by the labelling of the majority of the training examples that have reached the leaf as in ML-C4.5 [11]. In the ranking scenario, the labels are ordered according to their frequency among the training examples in the corresponding leaves.

Once a tree has been built, it is pruned back. The general algorithm to prune the tree is presented on Figure 3. The tree is traversed using a bottom-up, post order strategy. The pruning is conducted using a validation dataset (S^{val}). Each sample of the validation dataset traverses the tree until it reaches a leaf, where it is being labelled in the same way as in the ML-C4.5 approach. For each node, two quantities are estimated using the validation dataset: *leaf error* and *tree error*. *Leaf error* is an estimate of the error considering the node as a leaf. *Tree error* is defined as a weighted sum of the estimates of the errors of all subtrees of the current node. If the *tree error* is greater or equal than the *leaf error*, then the node is replaced by a leaf (subtree replacement [46]). The estimation of the error can be conducted using any performance evaluation measure (for example, evaluation measures described in section 4.1).

After pruning the tree, we try to solve the "new" classification problems defined on the examples in the leaves of the tree by using more accurate local models (Figure 4). This means that in each leaf of the decision tree one local model is learned with the examples from the corresponding leaf by the BR method, using SVMs as base classifiers for solving the partial binary classification problems. After the procedure of learning the local SVM models, the error of each local model is estimated by

procedure ML-SVMDT(S^{train}, S^{val}) **returns** tree
1: $tree$ = ML-DT(S^{train})
2: PruneByValidation($tree$, S^{val})
3: BuildLocalModels($tree$, S^{val})
4: **return** $tree$

Fig. 1. The procedure of building the ML-SVMDT.

using the subset of the validation dataset (S^{val}) that reached the leaf where the corresponding local model is located. If the *local model error* is smaller than the *leaf error* in a corresponding leaf, the examples from the training and validation datasets are joined first (line 7 on Figure 4), and then the leaf is replaced with the local model that is learned on the joined dataset. The last step (learning of the local models on the joined training and validation datasets) is employed in order to increase the predictive performance of the local models. The validation dataset is usually held out from the training dataset, so its examples are actually a part of the training dataset. The predictive performance should increase, as a result of learning the local models on the extended training subsets with examples that have similar properties to the original training examples.

Instead of using local models in every leaf of the tree, the local models in our architecture are employed only in the leaves where they showed lower error than the corresponding leaf of the decision tree in the validation phase. This means that not all leaves of the tree are replaced by local SVM models. Because of that, we expect the computational complexity of the architecture to be reduced in the prediction phase, as a result of the higher computational efficiency of the decision trees as compared to the SVM models.

The fundamental aspect of the pruning algorithm and the integration of the local SVM models in the decision tree is the error estimation (line 4 on Figure 3 and line 6 on Figure 4). The error depends on the loss-function used. Different loss-functions can be used for multi-label classification or multi-label ranking such as Hamming loss, Ranking loss, Accuracy, Coverage, etc., depending on the type of the problem that has to be solved.

Figure 5 shows an example of the ML-SVMDT architecture. The decision tree in the example is built according to the features f1 and f2. After splitting the dataset and pruning the tree, at the second level of the left subtree the local SVM model LSVM-2 is learned. In the left subtree of the second node (f2<6), another split is made using the feature f1 (f1<1) at the third level, resulting in another local SVM model (LSVM-1) and one leaf without a local SVM model (a leaf with a direct prediction). Similarly, in the right subtree the local SVM models LSVM-3 and LSVM-4 are learned. The squares, the circles, the crosses and the x-signs in the same Figure represent the distribution of the labels in the whole dataset. In combination with the separating lines, they represent the distribution of the labels

procedure ML-DT(S^{train}) **returns** tree

1: $(f^*, g^*, \mathcal{P}^*) = \text{BestFeature}(S^{train})$
2: **if** $f^* \neq none$ **then**
3: **for each** $S_k^{train} \in \mathcal{P}^*$ **do**
4: $tree_k = \text{ML-SVMDT}(S_k^{train})$
5: **return** node$(f^*, S^{train}, \bigcup_k \{tree_k\})$
6: **else**
7: **return** leaf(S^{train})

procedure BestFeature(S)

1: $(f^*, g^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: **for each** feature f **do**
3: $\mathcal{P}$ = partition induced by f on S
4: $g = \mathcal{E}(S) - \sum_{S_k \in \mathcal{P}} \frac{|S_k|}{|S|} \mathcal{E}(S_k)$
5: **if** $(g > g^*) \wedge \text{Acceptable}(\mathcal{P})$ **then**
6: $(f^*, g^*, \mathcal{P}^*) = (f, g, \mathcal{P})$
7: **return** $(f^*, g^*, \mathcal{P}^*)$

Fig. 2. The procedure of building the decision tree for multi-label classification.

procedure PruneByValidation($tree$, S^{val})

1: **if** $tree$ is not leaf **then**
2: **for each** descendent k **do**
3: PruneByValidation$(tree_k, S_k^{val})$
4: **if** LeafError$(tree, S^{val}) \leq$ TreeError$(tree, S^{val})$ **then**
5: $tree = \text{leaf}(tree.S^{train})$

Fig. 3. The process of validation-based post-pruning of the ML-SVMDT.

procedure BuildLocalModels($tree$, S^{val})

1: **if** $tree$ is not leaf **then**
2: **for** each descendent k **do**
3: BuildLocalModels$(tree_k, S_k^{val})$
4: **else**
5: $localSVM = \text{TrainLocalModel}(tree.S^{train})$
6: **if** LocalModelError$(localSVM, S^{val}) <$ LeafError$(tree, S^{val})$ **then**
7: $S = tree.S^{train} \cup S^{val}$
8: $localSVM = \text{TrainLocalModel}(S)$
9: $tree = \text{replaceNode}(localSVM)$

Fig. 4. The process of building of the local SVM models.

in the subsets (local datasets).

3.2. *Two modifications of ML-SVMDT*

In order to explore deeper the feasibility of the combination of the ML-C4.5 method and the local SVM models, two additional modifications of the ML-SVMDT architecture are proposed. The proposed modifications differ only in the second (pruning) and the third phase (integration of local SVM models), while the first phase (building the decision tree) is the same. The first modified architecture reffered to

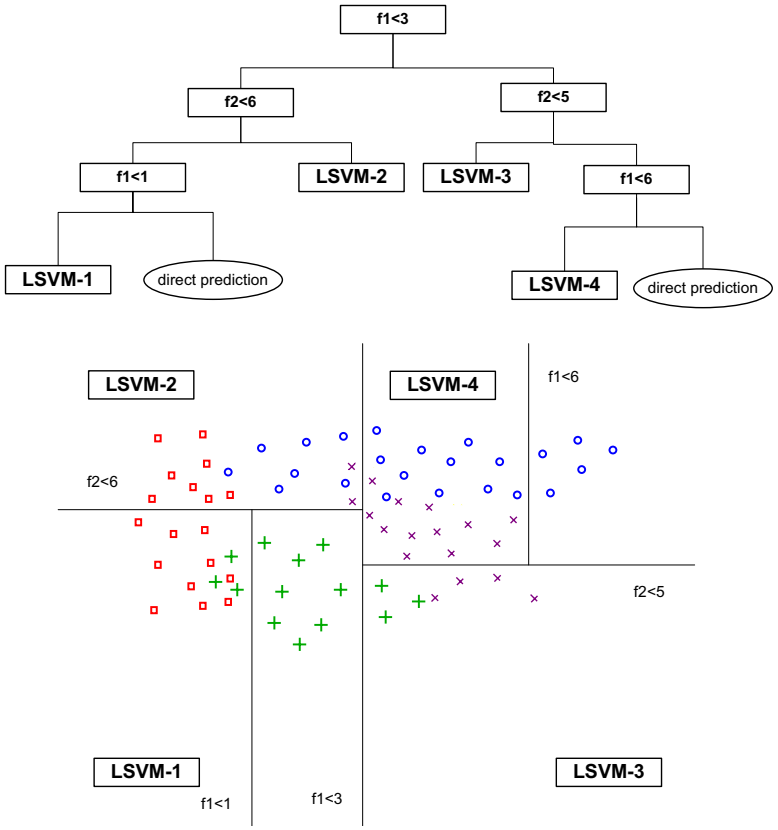

Fig. 5. ML-SVMDT splits the original dataset into subsets and builds a local SVM model (LSVM) for some partitions

as ML-SVMDT$_{\text{pre}}$ [47] does not include the pruning phase and employs the binary relevance architecture in each leaf of the decision tree, except in the leaves where all training examples are labelled with the same set of labels.

The second modification introduces a different pruning method as compared to the original ML-SVMDT architecture. Instead of using a validation dataset in the pruning phase, it uses subtree raising with a pruning confidence [46] as a pruning strategy. Similarly to ML-SVMDT$_{\text{pre}}$, after the pruning of the decision tree, it employees the binary relevance architecture with SVMs as base classifiers for solving local classification problems in the leaves of the decision tree. This modification is referred to as hybrid architecture with post-pruned decision tree (ML-SVMDT$_{\text{post}}$).

3.3. *The computational complexity of ML-SVMDT*

In this subsection, we analyse the computational complexity of the proposed method in comparison to the BR method. We first introduce the notation and then analyse the training and testing computational complexities separately.

The notation used in the analysis is: V is the number of leaves in the tree, d is the dimension of the input data (total number of descriptive attributes), while M is the number of continuous attributes. N represents the total number of training examples, while N^i represents the number of examples in the i-th leaf. N_{sv} is the number of support vectors in the global BR architecture (BR learned on the original non-split classification problem), while N_{sv}^i is the number of support vectors of the local SVM classifier in the i-th leaf.

3.3.1. *Training computational complexity*

It can be expected that the computational efficiency of the BR method will be improved as a result of splitting the original problem by the decision tree into smaller subproblems and as a result of introducing a local model for solving each of those subproblems. Each local model is learned with the examples that belong to one subset only and have smaller number of support vectors resulting in higher speed in the prediction phase.

The computational complexity in the training phase of the BR method according to Burges [48] and Bottou [49] depends on the examples in the training dataset, the dimension of the input data and on the number and the distribution of the support vectors determined in the learning process. In the case where most support vectors are not at the upper bound and $N_{sv}/N \ll 1$, the computational complexity is $\mathcal{O}({N_{sv}}^3 + {N_{sv}}^2 N + dN_{sv}N)$. If instead $N_{sv}/N \approx 1$, then the computational complexity is $\mathcal{O}({N_{sv}}^3 + N_{sv}N + dN_{sv}N)$. For the case where most SVs are at the upper bound and $N_{sv}/N \ll 1$, the computational complexity is $\mathcal{O}({N_{sv}}^2 + dN_{sv}N)$. Finally, if most SVs are at the upper bound and $N_{sv}/N \approx 1$, then the computational complexity is $\mathcal{O}(dN^2)$.

The computational complexity in the training phase of ML-SVMDT can be defined as a sum of the computational complexity of the ML-C4.5 method and the computational complexity of the local BR classifiers. The construction of the tree has three parts that contribute to the computational complexity of the tree learning algorithm described in the previous subsection. These parts are executed at each node in the tree and they include: sorting of the values of the numeric descriptive attributes (features), calculating the best split and applying the split to the training examples. Sorting a single descriptive attribute costs $\mathcal{O}(NlogN)$, thus sorting all numeric descriptive attributes costs $\mathcal{O}(MNlogN)$. The cost of passing all examples and calculating the needed statistics for all descriptive attributes is $\mathcal{O}(dN)$. Splitting the examples into the respective nodes costs $\mathcal{O}(N)$. To sum up, the computational complexity to creating a node in the tree is $\mathcal{O}(MNlogN) + \mathcal{O}(dN) + \mathcal{O}(N)$. If we assume that the tree is balanced and bushy as in [50], the total computational

complexity of tree construction is $\mathcal{O}_{ML-C4.5} = \mathcal{O}(MNlog^2N) + \mathcal{O}(dNlogN) + \mathcal{O}(NlogN)$. The upper bound of this cost is determined by the first term of the overall computational complexity, i.e., $\mathcal{O}(MNlog^2N)$.

The computational complexity of the local classifiers can be calculated similarly to the global classifier. For the i-th local classifier, in the case where most SVs are not at the upper bound and $N^i_{sv}/N^i \ll 1$, the average computational complexity is $\mathcal{O}({N^i_{sv}}^3 + {N^i_{sv}}^2 N^i + dN^i_{sv}N^i)$. If $N^i_{sv}/N^i \approx 1$, then the average computational complexity would be $\mathcal{O}({N^i_{sv}}^3 + N^i_{sv}N^i + dN^i_{sv}N^i)$. For the case where most SVs are at the upper bound and $N^i_{sv}/N^i \ll 1$, the computational complexity is $\mathcal{O}({N^i_{sv}}^2 + dN^i_{sv}N^i)$. If most SVs are at the upper bound and $N^i_{sv}/N^i \approx 1$, then the computational complexity is $\mathcal{O}(d{N^i}^2)$. Different local classifiers could have different computational complexities (according to the previous statements) and that depends on the local subproblems that should be solved. The total computational complexity of the local classifiers is a sum of the complexities of all local classifiers and it is same or higher order of the computational complexity of the decision tree.

In the worst case, the computational complexity of the proposed hybrid architecture is the same order as the computational complexity of the BR method. But, it could be expected that the computational complexity of ML-SVMDT compared to the computational complexity of BR should be lower for the number of leaves in the decision tree as a result of the lower computational complexity of the local subproblems that should be solved.

3.3.2. *Testing computational complexity*

The prediction phase of an example starts at the root of the tree. The decision tree transfers the example to exactly one leaf of the tree according to its features. The final decision about the labelling of the example is performed either by the leaf itself or by the local model consisting of SVMs in the corresponding leaf. Each example, in order to be classified consults at most one local model. Because each local model will deal with a localized subproblem, it can be anticipated that its computational complexity will be quite low, so, it can be expected that the testing speed of the architecture will also be improved. The testing time for each test example is the sum of the time needed to sort the example through the decision tree and if the corresponding leaf has a local model, the time needed for the local model to make a decision.

The computational complexity in the testing phase of the BR method according to Burges [48] and Bottou [49] for each test example is

$$\mathcal{O}_{BR} = \mathcal{O}(TN_{sv}) \tag{2}$$

where T is parameter representing the number of operations required to evaluate the kernel. For dot product and Gaussian kernels, T is proportional to the dimension of the data vector d.

In general, the computational complexity in the testing phase of the ML-SVMDT for a test sample that traverses from the root to the i-th leaf can be defined as a sum of the computational complexity of ML-C4.5 and the computational complexity of the local BR classifier in the i-th leaf:

$$\mathcal{O}^i_{ML-SVMDT} = \mathcal{O}(h_i) + \mathcal{O}(TN^i_{sv}) \tag{3}$$

where h_i is the lenght of the path to the i-th leaf in the tree and N^i_{sv} is the number of support vectors of the local SVM classifier in the i-th leaf. $\mathcal{O}(h_i)$ represents the computational complexity of the decisions in the nodes required for traversing the test example from the root to the i-th leaf.

For an average test example, that has a equal probability to end up in any leaf of the tree, the computational complexity of the ML-SVMDT will be:

$$\mathcal{O}_{ML-SVMDT} = \mathcal{O}(logN) + \mathcal{O}(T\overline{N^i_{sv}}) \tag{4}$$

where $\overline{N^i_{sv}} = \frac{1}{V}\sum_{i=0}^{i<V} N^i_{sv}$ and $logN$ represents the height of the tree presuming that the decision tree is balanced and bushy. The upper bound of the computational complexity of the ML-SVMDT is determined by the second term $\mathcal{O}(T\overline{N^i_{sv}})$. If we assume that $N_{sv} = k\overline{N^i_{sv}}$, one can conclude that the testing speed of the ML-SVMDT is about k times higher than the testing speed of the binary relevance method (equation 5):

$$\mathcal{O}_{BR} = k \cdot \mathcal{O}_{ML-SVMDT} \tag{5}$$

The parameter k strongly depends on the number and distribution of the examples in the global classification problem and the number and distribution of the examples in each local problem in the leaves. Statistically, presuming that the suport vectors are approximately evenly distributed through the local models, the value of the parameter k can be expected to be proportional to the number of leaves in the decision tree (V). An additional speed up in the prediction phase can be expected as a result of the absence of local models in some leaves of the tree where direct prediction is performed.

4. Experiments

The goal of the experiments is to answer the following questions:

(1) Is the ML-SVMDT approach more accurate and efficient than its constituents ML-C4.5 and BR?
(2) Does ML-SVMDT have better predictive performance than competing methods?

(3) Does ML-SVMDT have better computational complexity than competing methods?

In order to answer these questions, we evaluate the predictive performance and the computational complexity of the proposed and several competing methods on a selection of multi-label datasets that vary in terms of problem domain, number of labels and label cardinality. We first introduce the evaluation measures, then the datasets with their relevant statistics. Following this, we review the experimental methods compared, and setup employed. Finally we present the results.

4.1. *Evaluation measures*

Three example-based measures (Hamming loss, F_1 score and subset accuracy), four label-based measures (macro-averaged precision, macro-averaged recall, micro-averaged precision and micro-averaged recall) and two ranking-based measure (one error and average precision) [51] are used in this paper for evaluating the predictive performance of the compared methods. In the definitions below, $\mathcal{Y}_i$ denotes the set of true labels of example $\mathbf{x_i}$ and $h(\mathbf{x_i})$ denotes the set of predicted labels for the same example. All definitions refer to the multi-label setting.

4.1.1. *Example based measures*

Hamming loss evaluates how many times an example-label pair is misclassified, i.e., label not belonging to the example is predicted or a label belonging to the example is not predicted. The smaller the value of $hamming\_loss(h)$, the better the performance. The performance is perfect when $hamming\_loss(h) = 0$. This metric is defined as:

$$hamming\_loss(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{Q}\left|h(\mathbf{x_i})\Delta\mathcal{Y}_i\right| \tag{6}$$

where Δ stands for the symmetric difference between two sets, N is the number of examples and Q is the total number of possible class labels.

F_1 **score** is the harmonic mean between precision and recall and is defined as:

$$F_1 = \frac{1}{N}\sum_{i=1}^{N}\frac{2\times\left|h(\mathbf{x_i})\cap\mathcal{Y}_i\right|}{\left|h(\mathbf{x_i})\right|+\left|\mathcal{Y}_i\right|} \tag{7}$$

F_1 is an example based metric and its value is an average over all examples in the dataset. F_1 reaches its best value at 1 and worst score at 0.

Subset accuracy or classification accuracy is defined as follows:

$$subset\_accuracy(h) = \frac{1}{N}\sum_{i=1}^{N} I(h(\mathbf{x_i}) = \mathcal{Y}_i) \tag{8}$$

where $I(true) = 1$ and $I(false) = 0$. This is a very strict evaluation measure as it requires the predicted set of labels to be an exact match of the true set of labels.

4.1.2. *Label based measures*

Macro precision (precision averaged across all labels) is defined as:

$$macro\_precision = \frac{1}{Q}\sum_{j=1}^{Q}\frac{tp_j}{tp_j + fp_j} \tag{9}$$

where tp_j, fp_j are the number of true positives and false positives for the label λ_j considered as a binary class.

Macro recall (recall averaged across all labels) is defined as:

$$macro\_recall = \frac{1}{Q}\sum_{j=1}^{Q}\frac{tp_j}{tp_j + fn_j} \tag{10}$$

where tp_j, fp_j are defined as for the macro precision and fn_j is the number of false negatives for the label λ_j considered as a binary class.

Micro precision (precision averaged over all the example/label pairs) is defined as:

$$micro\_precision = \frac{\sum_{j=1}^{Q} tp_j}{\sum_{j=1}^{Q} tp_j + \sum_{j=1}^{Q} fp_j} \tag{11}$$

where tp_j, fp_j are defined as for macro precision.

Micro recall (recall averaged over all the example/label pairs) is defined as:

$$micro\_recall = \frac{\sum_{j=1}^{Q} tp_j}{\sum_{j=1}^{Q} tp_j + \sum_{j=1}^{Q} fn_j} \tag{12}$$

where tp_j and fn_j are defined as for macro recall.

4.1.3. *Ranking based measures*

One error evaluates how many times the top-ranked label is not in the set of relevant labels of the example. The metric $one\_error(f)$ takes values between 0 and 1. The smaller the value of $one\_error(f)$, the better the performance. This evaluation metric is defined as:

$$one\_error(f) = \frac{1}{N}\sum_{i=1}^{N}\left[\left[\arg\max_{\lambda\in\mathcal{Y}} f(\mathbf{x_i}, \lambda)\right] \notin \mathcal{Y}_i\right] \tag{13}$$

where $\lambda \in \mathcal{L} = \{\lambda_1, \lambda_2, ..., \lambda_Q\}$ and $[\![\pi]\!]$ equals 1 if π holds and 0 otherwise for any predicate π. Note that, for single-label classification problems, the One Error is identical to ordinary classification error.

Average Precision is the average fraction of labels ranked above an actual label $\lambda \in \mathcal{Y}_i$ that actually are in $\mathcal{Y}_i$. The performance is perfect when

$avg\_precision(f) = 1$; the larger the value of $avg\_precision(f)$, the better the performance. This measure is defined as:

$$avg\_precision(f) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\mathcal{Y}_i|}\sum_{\lambda\in\mathcal{Y}_i}\frac{|\mathcal{L}_i|}{rank_f(\mathbf{x_i},\lambda)} \tag{14}$$

where $\mathcal{L}_i = \{\lambda' | rank_f(\mathbf{x_i},\lambda') \leq rank_f(\mathbf{x_i},\lambda), \lambda' \in \mathcal{Y}_i\}$ and $rank_f(\mathbf{x_i},\lambda)$ maps the outputs of $f(\mathbf{x_i},\lambda)$ for any $\lambda \in \mathcal{L}$ to $\{\lambda_1,\lambda_2,...,\lambda_Q\}$ so that $f(\mathbf{x_i},\lambda_m) > f(\mathbf{x_i},\lambda_n)$ implies $rank_f(\mathbf{x_i},\lambda_m) < rank_f(\mathbf{x_i},\lambda_n)$.

4.2. *Compared methods*

The proposed architecture (ML-SVMDT) and its two modifications are compared with nine state of the art approaches for multi-label learning. The proposed architecture for the pruning phase and the phase of integrating the local SVM models with the decision tree uses a loss function that can be any performance evaluation measure. In our experiments we use Hamming loss as a loss function. For easy reference, Table 1 lists all the compared methods used in the experiments, their types and the relevant citations.

Table 1. Compared methods

name	**symbol**	**type**	**reference**
Proposed hybrid architecture with Hamming loss as a loss function	**ML-SVMDT$_{\mathbf{HL}}$**	hybrid	
Architecture with pre-pruning	**ML-SVMDT$_{\mathbf{pre}}$**	hybrid	
Architecture with post-pruning	**ML-SVMDT$_{\mathbf{post}}$**	hybrid	
Binary relevance	**BR**	binary relevance	[2]
Chaining classifier	**CC**	binary relevance	[18]
Calibrated label ranking	**CLR**	pair-wise	[24]
QWeightedML	**QWML**	pair-wise	[26]
HOMER	**HOMER**	problem transformation	[21]
ML-C4.5	**ML-C4.5**	algorithm adaptation	[11]
PCT	**PCT**	algorithm adaptation	[12]
ML-kNN	**ML-kNN**	algorithm adaptation	[3]
RAKEL	**RAKEL**	ensemble	[20]

4.3. *Datasets*

Eleven different multi-label classification problems were addressed in our experiments. The predictive performance in terms of the measures defined above and the

training and testing times were recorded for every method for each classification problem. The problems considered in the experiments include:

(1) image classification: scene [52] and corel5k [53];
(2) gene function classification: yeast [54];
(3) text classification: enron [55], medical [28], bibtex [39], delicious [21], bookmarks [39] and tmc2007 [56];
(4) music classification: emotions [57];
(5) video classification: mediamill [58]

The complete description of the datasets in terms of the number of training ($\#tr.e.$) and test ($\#t.e.$) examples, the number of features (d), the total number of labels (Q) and label cardinality (l_c) [2] are shown in Table 2.

Table 2. Dataset description.

	$\#tr.e.$	$\#t.e.$	d	Q	l_c
emotions	391	202	72	6	1.87
scene	1211	1159	294	6	1.07
yeast	1500	917	103	14	4.24
medical	645	333	1449	45	1.25
enron	1123	579	1001	53	3.38
corel5k	4500	500	499	374	3.52
tmc2007	21519	7077	500	22	2.16
mediamill	30993	12914	120	101	4.38
bibtex	4880	2515	1836	159	2.40
delicious	12920	3185	500	983	19.02
bookmarks	60000	27856	2150	208	2.03

We strived to include a considerable variety and scale of multi-label datasets. In total we use eleven datasets, with dimensions ranging from 6 to 983 labels, and from less than 1,000 examples to almost 90,000. The datasets are roughly ordered by complexity ($\#tr.e. \times d \times Q$) and divided in two groups denoted as regular size and large size datasets.

4.4. *Experimental setup*

The comparison of the multi-label learning methods was performed using the implementations in the following machine learning systems: the **MULAN**[a] [59] library under the machine learning framework WEKA [60], the **MEKA**[b] extension for the

[a]http://mulan.sourceforge.net/
[b]http://meka.sourceforge.net/

WEKA framework and **CLUS**[c] system for predictive clustering. The MULAN library was used for BR, CLR, QWML, HOMER, ML-C4.5, ML-kNN and RAkEL, while the MEKA environment was used for CC and the CLUS system for PCT. All experiments were performed on a server with an Intel Xeon processor at 2.50GHz with 64GB of RAM under the Fedora 14 operating system. In the remainder of this section, we first state the base classifiers that were used for the multi-label methods and then the parameter instantiations of the methods.

4.4.1. *Base classifiers*

The LIBSVM library [61], and in particular SVMs with a radial basis kernel, were used for solving the partial binary classification problems for all datasets in all problem transformation methods. The kernel parameter *gamma* and the penalty C, for each combination of dataset and method, were determined by 10-fold cross validation using only the training sets. The exception to this is the ensemble method RAkEL, where the kernel parameter *gamma* and the penalty C were determined by 5-fold cross validation for the tmc2007 and mediamill datasets because of the computational complexity. The values $2^{-15}, 2^{-13}, ..., 2^{1}, 2^{3}$ were considered for *gamma* and $2^{-5}, 2^{-3}, ..., 2^{13}, 2^{15}$ for the penalty C. After determining the best parameters values for each method on every dataset, the classifiers were trained using all available training examples and were evaluated by recognizing all test examples from the corresponding dataset.

4.4.2. *Parameter instantiation*

The parameters of the methods were instantiated following the recommendations from the literature. In particular, for the ensemble method (RAkEL) the number of models was set to $min(2 \cdot Q, 100)$ (Q is the number of labels) for all datasets [20], except for the mediamill, delicious and bookmarks datasets, where this parameter was set to 10 due to the memory requirements of this method. Besides the number of base classifiers, RAkEL requires one additional parameter: the size of the label-sets k. For each dataset, this parameter was set to half the number of labels ($Q/2$). Tsoumakas et al. [20] and Read et al. [28] have shown that this is a reasonable choice, since it provides a balance between computational complexity and predictive performance.

The proposed method uses one quarter of the training examples for validation, while the remaining examples are used for building the decision tree. The ML-C4.5 and ML-SVMDT$_{\text{post}}$ methods use subtree raising [46] as a post-pruning strategy with a pruning confidence set to 0.25. For the ML-SVMDT$_{\text{pre}}$ method we considered six different values (30-80) for the minimal number of examples in the leaves of the decision tree. PCTs use a pre-pruning strategy that employs the F-test to

[c]http://clus.sourceforge.net

determine whether a given split results in a significant reduction of variance. The significance level for the F-test was automatically selected from a predefined list of significance levels using 3-fold cross-validation. The number of neighbours in the ML-kNN method for each dataset was determined from the values 6 to 20 with step 2. HOMER also requires one additional parameter to be configured: the number of clusters. For this parameter, five different values (2-6) were considered in the experiments [21].

5. Results and discussion

In this section, we present the results from the experimental evaluation. The results achieved in the experiments on the regular datasets are shown first, followed by the results obtained on the large datasets. For each type of evaluation measure (Hamming loss, F_1 score, subset accuracy, micro precision, micro recall, macro precision, macro recall, one error and average precision), we present table(s) with results (Tables 3 to 5 and Tables 8 to 10). The training and testing times of each method on each of the datasets measured in seconds, are given in Table 6 and Table 11. The training time of the ML-kNN method is the time needed for calculating the posterior probabilities of the labels in the multi-label classification problems. For all methods, the best results are shown, achieved by selecting the optimal values for the corresponding parameters following the experimental setup explained in Section 4.4. The first column of the tables lists the evaluation measures, while the second column lists the classification problems. The remaining columns show the performance of each method for every dataset. The best results per dataset are shown in boldface.

For some of the large datasets, several algorithms did not manage to construct a predictive model within one week under the available resources. These occurrences are marked as DNF (Did Not Finish) in the tables with the results.

5.1. *Results on regular datasets*

Tables 3 to 6 give the performance figures for each method on each of the regular datasets measured in terms of the performance measures introduced in section 4.1, as well as the training and the testing times. In addition, Table 7 presents a statistic about the number of the leaves and local SVM models in the ML-SVMDT$_{\text{HL}}$ architecture for the regular datasets. The results in Tables 3 to 6 clearly show that among the twelve tested approaches, the proposed method ML-SVMDT$_{\text{HL}}$ and its modifications (ML-SVMDT$_{\text{pre}}$ and ML-SVMDT$_{\text{post}}$) offer better or show comparable predictive performance as compared to the other algorithms, for almost every evaluation measure.

Tables 3 and 5 clearly show that the ML-C4.5 gives the best results for the smallest dataset (emotions) for the example and ranking based measures. For the second smallest dataset (scene), the ensemble method (RAKEL) achieved the best performance for all but the average precision evaluation measure.

In terms of the Hamming loss evaluation measure ML-SVMDT$_{\mathrm{HL}}$ shows better predictive performance, than all the other competing methods in four out of five multi-label classification problems. For the second and third example based measures (F_1 score and subset accuracy), the best performance is achieved by ML-SVMDT$_{\mathrm{HL}}$ and HOMER.

For the label-based evaluation measures (micro and macro precision), SVM based approaches (BR, our proposed architecture and CC) achieved better overall results than the other competing methods. In terms of the recall-based evaluation measures (micro and macro recall), HOMER is best for all but the medical dataset. The best performance in terms of the ranking based evaluation measures is shown by the binary methods (BR, CC and CLR) and the proposed hybrid method ML-SVMDT$_{\mathrm{HL}}$.

Overall, the performance of ML-SVMDT$_{\mathrm{HL}}$ is always in the top four, compared to the competing methods in terms of the all evaluation measures for all regular datasets. It is interesting to note, that the proposed method almost always outperforms its constituents (BR and ML-C4.5) or shows very similar results to the better one.

As expected, PCT, ML-C4.5 and ML-kNN are the fastest methods in the training phase. ML-SVMDT$_{\mathrm{pre}}$ and ML-SVMDT$_{\mathrm{post}}$ have shorter training times compared to the other methods, while ML-SVMDT$_{\mathrm{HL}}$ has slightly longer training time as a result of the validation in the post pruning phase. In the testing phase, the proposed method achieve comparable testing times to ML-kNN and outperforms the other competing methods, except the PCT and ML-C4.5 methods that achieved the best testing times for all regular datasets. ML-SVMDT$_{\mathrm{HL}}$ is slightly faster in the testing phase than ML-SVMDT$_{\mathrm{pre}}$ and ML-SVMDT$_{\mathrm{post}}$ as a result of the absence of the local models in some leaves of the tree.

Although the predictive performances of the different methods on the regular size datasets are not that different, the training and testing times of the compared methods are spread over more than two orders of magnitude. ML-SVMDT$_{\mathrm{HL}}$ training and testing times are only slower than PCT, ML-C4.5 and ML-kNN (fastest but with worst predictive overall performance), while showing predictive performance that is very similar to, and often better than the other methods.

5.2. *Results on large datasets*

Tables 8 to 11 give the performance figures for each method on each of the large datasets measured in terms of the performance measures introduced in section 4.1, as well as the training and the testing times. Table 12 shows the number of leaves and local SVM models in the ML-SVMDT architecture for the large datasets.

In this subsection, we discuss the computational efficiency of the compared methods first, and then we analyse their predictive performances. Overall, the proposed method (ML-SVMDT$_{\mathrm{HL}}$) is 1.2 to 30 times faster in the training phase and 1.4 to 30 times faster in the testing phase than the HOMER method. The computa-

Table 3. The performance of the multi-label classification approaches on the regular datasets in terms of the examples-based measures.

	Dataset	ML-SVMDT$_{HL}$	ML-SVMDT$_{pre}$	ML-SVMDT$_{post}$	BR	CC	CLR	QWML	HOMER	ML-C4.5	PCT	ML-kNN	RAkEL
Hamming loss	**emotions**	**0.247**	0.254	0.257	0.257	0.256	0.257	0.254	0.361	**0.247**	0.267	0.294	0.282
	scene	**0.077**	0.097	0.099	0.079	0.082	0.080	0.081	0.082	0.141	0.129	0.099	**0.077**
	yeast	**0.190**	0.205	0.204	**0.190**	0.193	**0.190**	0.191	0.207	0.234	0.219	0.198	0.192
	medical	**0.011**	**0.011**	**0.011**	**0.011**	**0.011**	0.017	0.012	0.012	0.013	0.023	0.017	0.012
	enron	0.046	0.049	0.050	**0.045**	0.064	0.048	0.048	0.051	0.053	0.058	0.051	**0.045**
	avg. rank	1.4	5.4	6.2	2.4	6	5	4.4	8.4	8.8	10.8	9	4.6
F_1 score	**emotions**	0.614	0.563	0.545	0.469	0.461	0.465	0.481	0.614	**0.651**	0.554	0.431	0.525
	scene	0.714	0.661	0.655	0.714	0.742	0.713	0.710	0.745	0.587	0.551	0.658	**0.754**
	yeast	0.623	0.634	0.638	0.650	0.657	0.655	0.654	**0.687**	0.614	0.578	0.628	0.661
	medical	0.779	0.775	**0.781**	0.744	0.745	0.742	0.745	0.761	0.768	0.253	0.560	0.704
	enron	0.602	0.563	0.556	0.582	0.484	0.600	0.525	**0.613**	0.546	0.295	0.445	0.564
	avg. rank	4	5.8	6.2	6.2	6.6	6.4	7	2.2	7	10.6	10.4	5
Subset Accuracy	**emotions**	0.183	0.178	0.198	0.129	0.124	0.144	0.149	0.163	**0.277**	0.223	0.084	0.208
	scene	0.639	0.590	0.585	0.639	0.685	0.633	0.630	0.661	0.533	0.509	0.573	**0.694**
	yeast	0.168	0.165	0.166	0.190	**0.239**	0.195	0.192	0.213	0.158	0.152	0.159	0.201
	medical	**0.655**	0.643	**0.655**	0.630	0.621	0.486	0.480	0.610	0.646	0.177	0.462	0.607
	enron	0.145	0.140	0.138	**0.149**	0.000	0.117	0.097	0.145	0.140	0.002	0.062	0.136
	avg. rank	3.8	6.2	5.6	6.4	7.4	6.8	7.4	3.8	6	9.4	10.2	4

tional efficiency of ML-SVMDT$_{\mathrm{HL}}$ when compared to the BR and the CC methods is even higher (1.2 to 42 times in the training phase and 3 to 40 times in the testing phase). The other competing methods, except ML-kNN, show even higher training and testing times than the binary relevance methods (BR and CC). The pairwise methods (CLR and QWML) could not be learned within the training time frame with the two most complex datasets (delicious and bookmarks), while the ensemble method (RAKEL) was learned only for the corel5k, tmc2007 and mediamill datasets. Also, the learning phase of HOMER did not finish for the bookmarks dataset within one week under the available resources. Because of that the testing phase for these methods was not performed at all for the corresponding datasets.

As expected, ML-SVMDT$_{\mathrm{pre}}$ and ML-SVMDT$_{\mathrm{post}}$ are slightly faster in the training phase compared to ML-SVMDT$_{\mathrm{HL}}$, while in the testing phase these two modifications are slower. PCT and ML-C4.5 are faster than ML-SVMDT$_{\mathrm{HL}}$ in the training phase and they are the fastest in the testing phase for all datasets. It is interesting to note that ML-SVMDT$_{\mathrm{pre}}$ shows shorter training times compared to ML-C4.5 for the corel5k and delicious datasets (the datasets with the largest number of labels), as a result of the post-pruning method used in the ML-C4.5 algorithm that adds to its computational complexity in the training phase. On the other hand, ML-SVMDT$_{\mathrm{pre}}$ uses only the minimal number of examples in the leaves of the tree that controls the size of the tree - after a node reaches the minimal number of examples, no further branching of the decision tree is allowed. The

Table 4. The performance of the multi-label classification approaches on the regular datasets in terms of the label-based measures.

	Dataset	ML-SVMDT$_{HL}$	ML-SVMDT$_{pre}$	ML-SVMDT$_{post}$	BR	CC	CLR	QWML	HOMER	ML-C4.5	PCT	ML-kNN	RAkEL
Micro precision	**emotions**	0.683	0.634	0.628	0.684	**0.698**	0.685	0.680	0.471	0.607	0.607	0.584	0.586
	scene	**0.843**	0.787	0.782	**0.843**	0.814	0.835	0.832	0.804	0.619	0.512	0.691	0.831
	yeast	**0.738**	0.695	0.698	0.733	0.726	0.729	0.727	0.647	0.618	0.698	0.736	0.720
	medical	0.843	0.834	0.830	0.858	0.851	0.669	0.667	0.807	0.796	0.826	0.807	**0.881**
	enron	0.669	0.668	0.657	0.721	0.492	0.652	0.687	0.597	0.613	0.601	0.684	**0.743**
	avg. rank	3	7	7.4	2.2	5.6	5.6	5.8	9.8	10	9	7	4.8
Macro precision	**emotions**	0.654	0.625	0.596	**0.721**	0.581	0.677	0.660	0.464	0.602	0.628	0.518	0.547
	scene	**0.844**	0.785	0.777	**0.844**	0.817	0.835	0.832	0.807	0.635	0.682	0.784	0.835
	yeast	0.559	0.515	0.545	**0.628**	0.602	0.614	0.614	0.471	0.377	0.479	0.600	0.480
	medical	0.364	0.379	0.320	**0.410**	0.395	0.288	0.285	0.287	0.263	0.018	0.267	0.269
	enron	0.246	0.266	**0.307**	0.258	0.260	0.205	0.242	0.241	0.142	0.023	0.170	0.222
	avg. rank	4	5.4	6.2	1.6	4.8	4.4	4.8	8.8	10.6	10	9	7.8
Micro recall	**emotions**	0.562	0.539	0.534	0.406	0.393	0.409	0.414	**0.782**	0.712	0.539	0.376	0.489
	scene	0.698	0.638	0.631	0.694	0.708	0.695	0.687	**0.727**	0.570	0.521	0.634	0.721
	yeast	0.587	0.577	0.579	0.587	0.588	0.595	0.595	**0.702**	0.603	0.492	0.543	0.602
	medical	0.730	0.735	0.738	0.725	0.727	0.782	**0.830**	0.742	0.720	0.227	0.522	0.600
	enron	0.529	0.457	0.457	0.464	0.472	0.532	0.541	**0.585**	0.440	0.246	0.353	0.435
	avg. rank	4.8	6.8	7.2	7.4	6.4	4.6	4.4	1.4	6.6	10.4	10.8	6.4
Macro recall	**emotions**	0.556	0.511	0.503	0.378	0.364	0.381	0.398	**0.775**	0.702	0.533	0.334	0.462
	scene	0.703	0.646	0.639	0.703	0.716	0.704	0.701	**0.734**	0.573	0.529	0.647	0.727
	yeast	0.343	0.344	0.348	0.355	0.357	0.361	0.361	**0.466**	0.375	0.269	0.308	0.352
	medical	0.323	0.313	0.339	0.423	**0.428**	0.307	0.324	0.282	0.249	0.022	0.163	0.183
	enron	0.128	0.119	0.130	0.120	0.146	0.139	0.120	**0.163**	0.107	0.030	0.075	0.097
	avg. rank	5.6	7.4	6.2	5.8	4.4	5.2	5.6	2.4	6.6	10.4	10.6	7.2

Table 5. The performance of the multi-label classification approaches on the regular datasets in terms of the ranking-based measures.

	Dataset	ML-SVMDT$_{HL}$	ML-SVMDT$_{pre}$	ML-SVMDT$_{post}$	BR	CC	CLR	QWML	HOMER	ML-C4.5	PCT	ML-kNN	RAkEL
One error	**emotions**	0.376	0.371	0.396	0.386	0.376	0.391	0.391	0.411	**0.347**	0.386	0.406	0.396
	scene	**0.180**	0.259	0.259	**0.180**	0.204	0.190	0.193	0.216	0.394	0.389	0.242	0.197
	yeast	0.267	0.253	0.246	0.236	0.268	**0.229**	0.233	0.248	0.312	0.264	0.234	0.254
	medical	0.153	0.156	0.162	**0.138**	**0.138**	0.168	0.165	0.216	0.198	0.612	0.279	0.312
	enron	0.237	0.264	0.273	0.237	0.238	**0.231**	0.269	0.314	0.309	0.392	0.280	0.290
	avg. rank	3.8	5.4	7	2.6	5	3.8	5	9	8.6	9.8	8	8.4
Avg. precision	**emotions**	0.721	0.747	0.742	0.721	0.724	0.718	0.679	0.698	**0.759**	0.713	0.694	0.713
	scene	**0.893**	0.843	0.840	**0.893**	0.881	0.886	0.864	0.848	0.751	0.745	0.851	0.862
	yeast	0.760	0.746	0.749	**0.768**	0.755	**0.768**	0.698	0.740	0.706	0.724	0.758	0.715
	medical	0.872	0.872	0.871	**0.893**	**0.893**	0.864	0.862	0.786	0.823	0.522	0.784	0.676
	enron	0.681	0.668	0.658	0.693	0.695	**0.699**	0.604	0.604	0.629	0.546	0.635	0.522
	avg. rank	3.2	5.2	6	2.2	3.2	3.6	9	8.8	7.8	10.4	7.8	9.4

Table 6. The training and the testing times of the multi-label classification approaches on the regular datasets measured in seconds.

	Dataset	ML-SVMDT$_{HL}$	ML-SVMDT$_{pre}$	ML-SVMDT$_{post}$	BR	CC	CLR	QWML	HOMER	ML-C4.5	PCT	ML-kNN	RAkEL
training times	**emotions**	4.0	2.0	2.0	4.0	6.0	10.0	10.0	4.0	0.3	**0.1**	0.4	5.0
	scene	29.0	25.0	25.0	71.0	99.0	195.0	195.0	68.0	8.0	**2.0**	14.0	79.0
	yeast	86.0	42.0	49.0	145.0	206.0	672.0	672.0	101.0	14.0	**1.5**	8.2	157.0
	medical	14.0	8.0	8.0	18.0	28.0	40.0	40.0	16.0	3.0	**0.6**	1.0	82.0
	enron	97.0	77.0	70.0	318.0	440.0	971.0	971.0	158.0	15.0	**1.1**	6.0	493.0
	avg. rank	6	4.2	4.2	7.6	9.6	10.8	10.8	6.8	2.6	1	2.4	9.8
testing times	**emotions**	0.4	0.6	0.6	1.0	1.0	3.0	2.0	1.0	**0.0**	**0.0**	0.4	2.0
	scene	8.0	12.0	13.0	25.0	25.0	87.0	40.0	21.0	1.0	**0.0**	14.0	72.0
	yeast	4.0	10.0	11.0	23.0	25.0	153.0	64.0	17.0	0.1	**0.0**	5.0	70.0
	medical	0.7	1.8	1.7	4.0	6.0	90.0	25.0	1.5	0.1	**0.0**	0.2	24.0
	enron	7.0	18.0	16.0	50.0	53.0	634.0	174.0	22.0	0.2	**0.0**	3.0	153.0
	avg. rank	3.4	5.4	5.4	7.8	8.4	12	10.4	6.6	1.8	1	3.8	10.4

Table 7. Number of leaves and local SVMs in the ML-SVMDT architecture for the regular datasets.

		emotions	scene	yeast	medical	enron
ML-SVMDT$_{HL}$	**Number of leaves**	15	37	22	11	41
	Numer of local models	8	20	7	6	21

training and testing times of ML-kNN are comparable to the proposed method, but the predictive performance of ML-kNN is almost always lower than the predictive performance of ML-SVMDT$_{\mathrm{HL}}$.

Overall, it can be noted that four groups of algorithms are clearly separated in terms of time efficiency. The first group contains the decision tree based methods (PCT and ML-C4.5). The second contains the proposed method and its modifications and the algorithm adaptation method ML-kNN. The third group contains the binary methods BR and CC, and the HOMER method. The pair-wise methods CLR and QWML and the ensemble method RAKEL are members of the last group. The ML-C4.5 and PCT (first group) have significantly higher computational efficiency as compared to the methods of the other three groups, but they show significantly lower predictive performance in terms of all evaluation measures. The methods of the second group show significantly higher computational efficiency, but similar predictive performance to the methods of the third group. Methods of the fourth group have the lowest computational efficiency. They have also showed worse predictive performance than methods of the second and the third group.

As the complexity of the classification problems increases, the grouping of the methods becomes even more evident, which is visually presented on Figures 6, 7

Table 8. The performance of the multi-label classification approaches on the large datasets in terms of the examples-based measures.

	Dataset	ML-SVMDT$_{HL}$	ML-SVMDT$_{pre}$	ML-SVMDT$_{post}$	BR	CC	CLR	QWML	HOMER	ML-C4.5	PCT	ML-kNN	RAkEL
Hamming loss	**corel5k**	**0.009**	**0.009**	**0.009**	0.017	0.017	0.012	0.012	0.012	0.010	**0.009**	**0.009**	**0.009**
	tmc2007	0.013	**0.011**	**0.011**	0.013	0.013	0.014	0.014	0.015	0.093	0.075	0.058	0.021
	mediamill	**0.030**	0.032	0.032	0.032	0.032	0.043	0.043	0.038	0.044	0.034	0.031	0.035
	bibtex	**0.011**	0.012	0.012	0.012	0.012	0.012	0.012	0.014	0.016	0.014	0.014	DNF
	delicious	**0.017**	0.018	0.018	0.018	0.018	DNF	DNF	0.022	0.019	0.019	0.018	DNF
	bookmarks	**0.008**	0.009	0.009	DNF	DNF	DNF	DNF	DNF	0.009	0.009	0.009	DNF
	avg. rank	1.3	1.8	1.8	4.7	4.7	7.2	7.2	8.2	8.5	6.0	4.2	7.8
F_1 score	**corel5k**	0.145	0.076	0.079	0.047	0.048	**0.293**	0.292	0.280	0.003	0.000	0.021	0.000
	tmc2007	0.936	**0.949**	**0.949**	0.934	0.939	0.933	0.933	0.934	0.126	0.554	0.699	0.904
	mediamill	0.562	0.557	0.556	0.557	0.539	0.134	0.135	**0.579**	0.054	0.490	0.570	0.471
	bibtex	**0.438**	0.390	0.400	0.433	0.434	0.417	0.421	0.426	0.117	0.069	0.174	DNF
	delicious	0.322	0.240	0.240	0.230	0.225	DNF	DNF	**0.343**	0.001	0.001	0.017	DNF
	bookmarks	0.262	0.262	**0.267**	DNF	DNF	DNF	DNF	DNF	0.257	0.135	0.213	DNF
	avg. rank	2.7	4	3.8	5.3	5.3	7	6.8	3.5	9.3	9.2	7	9.7
Subset Accuracy	**corel5k**	0.008	0.008	0.008	0.000	0.000	0.010	**0.012**	0.002	0.000	0.000	0.000	0.000
	tmc2007	0.776	0.796	**0.802**	0.772	0.787	0.767	0.768	0.765	0.078	0.215	0.305	0.734
	mediamill	0.092	0.091	0.091	0.080	0.080	0.044	0.044	0.053	0.049	0.065	**0.110**	0.060
	bibtex	0.183	0.179	0.179	0.194	**0.202**	0.183	0.186	0.165	0.095	0.004	0.056	DNF
	delicious	0.002	0.004	0.005	0.004	**0.006**	DNF	DNF	0.001	0.001	0.001	0.003	DNF
	bookmarks	0.205	0.210	**0.215**	DNF	DNF	DNF	DNF	DNF	0.209	0.129	0.187	DNF
	avg. rank	3.8	3.2	2.7	4.8	4.0	6.8	6.3	7.5	8.0	8.2	6.3	8.8

and 8. The figures show the relationship between the training and testing times and the predictive performance of the competing methods for Hamming loss, F_1 score and micro recall evaluation measures for mediamill, tmc2007 (the two most complex datasets, for which all methods have managed to build a model), and the bibtex dataset (for which, the predictions of the RAkEL method are only missing). Very similar relationships can be observed for the other evaluation measures and datasets.

The scatter plots on Figures 6 to 8 present the superiority of the proposed method that is always positioned near the top of the y axes (higher performance evaluation measure) and having only the much less accurate methods on its left side. The methods that are positioned on its right side show comparable predictive performance, but are an order of magnitude slower.

6. Conclusions

We propose a novel hybrid architecture that integrates Decision Trees and Support Vector Machines for computationally efficient multi-label learning. The architecture combines the algorithm adaptation method ML-C4.5 and the problem transformation method Binary Relevance, that uses SVMs as base classifiers for solving the

Table 9. The performance of the multi-label classification approaches on the large datasets in terms of the label-based measures.

	Dataset	ML-SVMDT$_{HL}$	ML-SVMDT$_{pre}$	ML-SVMDT$_{post}$	BR	CC	CLR	QWML	HOMER	ML-C4.5	PCT	ML-kNN	RAkEL
Micro precision	**corel5k**	0.698	0.664	0.662	0.061	0.061	0.338	0.339	0.308	0.160	0.000	**0.730**	0.000
	tmc2007	0.942	0.947	**0.951**	0.947	0.948	0.940	0.941	0.922	0.940	0.689	0.757	0.938
	mediamill	**0.755**	0.749	0.745	0.742	0.753	0.582	0.580	0.569	0.597	0.743	0.739	0.725
	bibtex	0.787	0.789	0.764	0.753	0.744	0.734	0.736	0.547	0.359	**1.000**	0.819	DNF
	delicious	**0.692**	0.662	0.641	0.658	0.660	DNF	DNF	0.396	0.000	0.000	0.651	DNF
	bookmarks	0.798	0.855	0.755	DNF	DNF	DNF	DNF	DNF	0.632	**0.947**	0.850	DNF
	avg. rank	2.8	2.7	4.2	5.8	5	7.8	7.5	8.8	8.2	6.3	4.8	9.2
Macro precision	**corel5k**	0.055	0.055	0.055	0.052	0.053	**0.059**	**0.059**	0.044	0.004	0.000	0.031	0.000
	tmc2007	0.980	**0.984**	0.983	0.972	0.972	0.964	0.965	0.954	0.925	0.386	0.780	0.973
	mediamill	0.261	0.258	0.201	0.112	0.144	0.140	0.133	0.107	0.046	**0.401**	0.308	0.025
	bibtex	0.513	0.495	0.520	0.528	**0.539**	0.503	0.490	0.391	0.128	0.006	0.192	DNF
	delicious	**0.319**	0.312	0.290	0.299	0.303	DNF	DNF	0.154	0.000	0.000	0.134	DNF
	bookmarks	**0.526**	0.485	0.485	DNF	DNF	DNF	DNF	DNF	0.292	0.018	0.414	DNF
	avg. rank	2.5	3	3.3	5.7	4.7	6	6.3	8	9	8.2	7	9
Micro recall	**corel5k**	0.109	0.055	0.058	0.057	0.057	0.258	**0.290**	0.248	0.002	0.000	0.015	0.000
	tmc2007	0.927	**0.943**	0.940	0.917	0.924	0.920	0.921	0.932	0.073	0.454	0.621	0.847
	mediamill	0.428	0.416	0.411	0.415	0.385	0.066	0.429	**0.537**	0.004	0.351	0.432	0.315
	bibtex	0.296	0.281	0.289	0.328	0.335	0.322	0.341	**0.353**	0.053	0.057	0.118	DNF
	delicious	0.182	0.148	0.149	0.143	0.144	DNF	DNF	**0.297**	0.000	0.000	0.101	DNF
	bookmarks	0.176	0.170	**0.178**	DNF	DNF	DNF	DNF	DNF	0.170	0.135	0.135	DNF
	avg. rank	3.7	4.8	4.2	6.2	5.7	6.7	4.5	2.7	9.3	9.0	7.0	9.5
Macro recall	**corel5k**	0.020	0.014	0.014	0.023	0.023	0.039	0.039	**0.041**	0.005	0.000	0.006	0.000
	tmc2007	0.906	0.903	0.902	0.915	**0.924**	0.914	0.914	0.897	0.085	0.235	0.418	0.739
	mediamill	0.062	0.060	0.061	0.049	0.044	0.028	0.028	0.074	0.002	0.029	**0.088**	0.020
	bibtex	0.223	0.198	0.203	0.250	**0.257**	0.236	0.238	0.247	0.034	0.006	0.049	DNF
	delicious	0.072	0.066	0.064	0.072	0.075	DNF	DNF	**0.103**	0.000	0.000	0.039	DNF
	bookmarks	0.095	0.103	**0.111**	DNF	DNF	DNF	DNF	DNF	0.098	0.016	0.070	DNF
	avg. rank	4.5	5.5	5.3	4	3.7	5.7	5.5	3.7	9.2	9.2	6.8	9.7

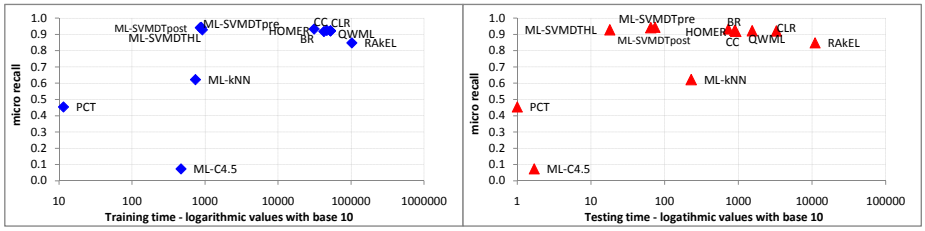

Fig. 6. The relationship between the training (left) and testing (right) times and the micro recall performance of the competing methods on the tmc2007 dataset.

partial binary classification problems.

The proposed architecture and its two modifications are compared to nine state

Table 10. The performance of the multi-label classification approaches on the large datasets in terms of the ranking-based measures.

	Dataset	ML-SVMDT$_{HL}$	ML-SVMDT$_{pre}$	ML-SVMDT$_{post}$	BR	CC	CLR	QWML	HOMER	ML-C4.5	PCT	ML-kNN	RAkEL
One error	**corel5k**	0.655	0.652	0.666	0.660	0.674	**0.588**	0.592	0.652	0.762	0.776	0.706	0.758
	tmc2007	0.021	**0.013**	0.016	0.029	0.026	0.033	0.033	0.050	0.145	0.306	0.190	0.047
	mediamill	**0.180**	0.183	0.184	0.188	0.193	0.586	0.560	0.219	0.194	0.220	0.182	0.234
	bibtex	0.384	0.407	0.382	0.346	**0.342**	0.388	0.380	0.466	0.529	0.783	0.576	DNF
	delicious	0.375	0.372	0.376	**0.354**	0.367	DNF	DNF	0.509	0.411	0.592	0.416	DNF
	bookmarks	0.599	0.594	**0.577**	DNF	DNF	DNF	DNF	DNF	0.643	0.817	0.639	DNF
	avg. rank	3.5	3.2	3.8	4.3	4.7	7	6.5	7.2	8	9.8	7.2	9.5
Avg. precision	**corel5k**	0.312	0.306	0.304	0.303	0.293	**0.352**	0.311	0.222	0.196	0.208	0.266	0.088
	tmc2007	0.979	**0.985**	0.983	0.978	0.981	0.972	0.938	0.945	0.842	0.700	0.844	0.939
	mediamill	**0.703**	0.698	0.697	0.686	0.672	0.450	0.492	0.583	0.669	0.654	**0.703**	0.492
	bibtex	0.551	0.536	0.557	0.597	**0.599**	0.579	0.498	0.407	0.392	0.212	0.349	DNF
	delicious	**0.364**	0.362	0.362	0.351	0.343	DNF	DNF	0.231	0.321	0.206	0.326	DNF
	bookmarks	0.420	0.421	**0.435**	DNF	DNF	DNF	DNF	DNF	0.378	0.213	0.381	DNF
	avg. rank	2.7	3.0	3	4.8	4.8	6.5	7.7	8	8.3	9.3	6.5	9.8

Table 11. The training and the testing times of the multi-label classification approaches on the large datasets measured in seconds.

	Dataset	ML-SVMDT$_{HL}$	ML-SVMDT$_{pre}$	ML-SVMDT$_{post}$	BR	CC	CLR	QWML	HOMER	ML-C4.5	PCT	ML-kNN	RAkEL
training times	**corel5k**	732.0	274.0	645.0	926.0	1225.0	2388.0	2388.0	771.0	369.0	**30.0**	389.0	3380.0
	tmc2007	920.0	888.0	862.0	42645.0	46704.0	52427.0	52427.0	31300.0	469.0	**11.5**	737.0	102394.0
	mediamill	10345.0	9015.0	10789.0	85468.0	100435.0	260156.0	260156.0	78195.0	2030.0	**440.0**	1094.0	33554.0
	bibtex	1304.0	767.0	769.0	11013.0	12434.0	13424.0	13424.0	2896.0	566.0	**16.4**	124.0	DNF
	delicious	1945.0	1168.0	1358.0	57053.0	84903.0	DNF	DNF	21218.0	2738.0	**70.0**	236.0	DNF
	bookmarks	160981.0	53737.0	137660.0	DNF	DNF	DNF	DNF	DNF	4039.0	**965.0**	15990.0	DNF
	avg. rank	5.7	3.7	4.8	8	8.8	9.7	9.7	7.2	3.2	1	2.7	10.0
testing times	**corel5k**	8.0	9.0	14.0	25.0	31.0	2161.0	119.0	14.0	**1.0**	**1.0**	45.0	3613.0
	tmc2007	18.0	74.0	65.0	927.0	891.0	3282.0	1543.0	730.0	1.7	**0.0**	230.0	10985.0
	mediamill	353.0	398.0	470.0	6152.0	6125.0	76385.0	20317.0	6079.0	**1.0**	**1.0**	477.0	39001.0
	bibtex	48.0	84.0	84.0	654.0	661.0	16733.0	4710.0	155.0	6.5	**0.0**	64.0	DNF
	delicious	102.0	189.0	182.0	2045.0	1872.0	DNF	DNF	816.0	19.0	**10.0**	55.0	DNF
	bookmarks	1480.0	4189.0	8022.0	DNF	DNF	DNF	DNF	DNF	21.0	**15.0**	4084.0	DNF
	avg. rank	3.2	4.8	5	8	8.2	10.2	9.3	6.7	1.7	1	5.3	10.5

Table 12. Number of leaves and local SVM models in the ML-SVMDT architecture for the large datasets.

		corel5k	tmc2007	mediamill	bibtex	delicious	bookmarks
ML-SVMDT$_{HL}$	**Number of leaves**	23	38	61	24	44	130
	Numer of local models	21	36	49	16	37	74

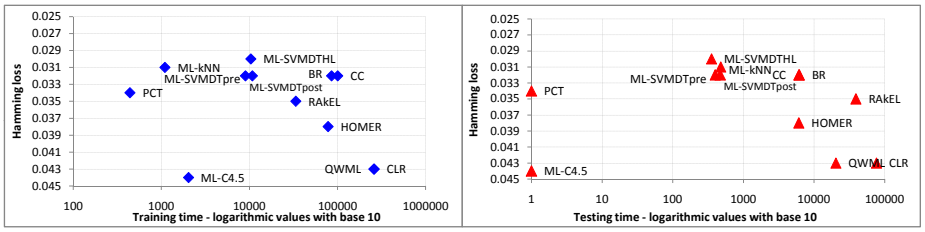

Fig. 7. The relationship between the training (left) and testing (right) times and the Hamming loss of the competing methods on the mediamill dataset.

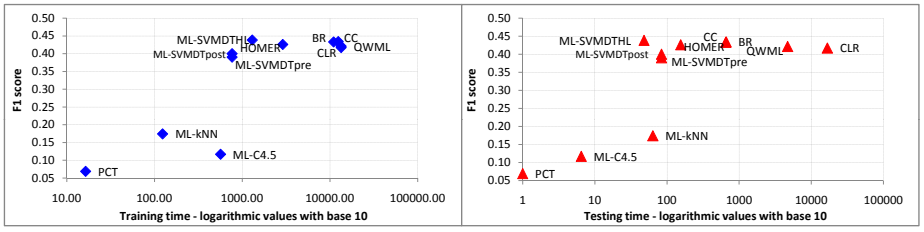

Fig. 8. The relationship between the training (left) and testing (right) times and the F_1 score of the competing methods on the bibtex dataset.

of the art multi-label methods (five problem transformation methods, three algorithm adaptation methods and one ensemble method) on eleven different real-world datasets separated in two groups according to their complexity.

Overall, the proposed architecture shows significantly smaller training and testing times in comparison to the SVM-based methods. It offers better or shows comparable predictive performance in terms of the performance evaluation measures to the best results achieved by the competing methods. Compared to the BR and CC methods, ML-SVMDT architecture shows slightly better predictive performance, but significantly higher computational efficiency, especially for the large datasets. Also, the architecture outperforms the HOMER method in both training and testing speed and ranking. In comparison to non SVM-based methods, ML-SVMDT shows significantly better predictive performance than PCT, ML-C4.5 and ML-kNN methods in terms of the nine evaluation measures, while showing higher training and testing times. Compared to other methods, ML-SVMDT offers competitive predictive performance and is efficient enough to scale up to very large problems.

Despite the above advantages, the method we proposed might perform below expectation in a couple of situations. For example, on datasets that yield highly imbalanced decision trees, the computational efficiency of our approach will be low, but not lower than the efficiency of the baseline BR approach. Another drawback of our method is its high memory consumption for large datasets: For the most complex datasets (bibtex, bookmarks...), the process of training takes about 40GB

of RAM for all local SVM models.

This leads us to the directions for further work. Our approach could be easily parallelized. Besides speeding up the training process, this would reduce the memory requirements.

Several other directions can also be followed. Statistical tests could be used to decide whether further splitting makes sense, instead of using a post-pruning strategy with validation. Different ensemble techniques such as random forest, bagging or boosting can be introduced in order to improve the predictive performance of the hybrid architecture.

References

1. K. Brinker, J. Fürnkranz, and E. Hüllermeier, "A unified model for multilabel classification and ranking," in *Proc. of the 17th European Conference on Artificial Intelligence*, pp. 489–493, 2006.
2. G. Tsoumakas and I. Katakis, "Multi Label Classification: An Overview," *International Journal of Data Warehouse and Mining*, vol. 3, no. 3, pp. 1–13, 2007.
3. M. L. Zhang and Z. H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.
4. A. Wieczorkowska, P. Synak, and Z. Ras, "Multi-label classification of emotions in music," in *Intelligent Information Processing and Web Mining*, pp. 307–315, Springer Berlin / Heidelberg, 2006.
5. E. Spyromitros, G. Tsoumakas, and I. Vlahavas, "An empirical study of lazy multilabel classification algorithms," in *Proc. of the 5th Hellenic conference on Artificial Intelligence: Theories, Models and Applications*, pp. 401–406, 2008.
6. K. Crammer and Y. Singer, "A family of additive online algorithms for category ranking," *Journal of Machine Learning Research*, vol. 3, pp. 1025–1058, 2003.
7. M. L. Zhang and Z. H. Zhou, "Multi-label neural networks with applications to functional genomics and text categorization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1338–1351, 2006.
8. R. E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," *Machine Learning*, vol. 39, pp. 135–168, 2000.
9. F. De Comité, R. Gilleron, and M. Tommasi, "Learning multi-label alternating decision trees from texts and data," in *Proc. of the 3rd international conference on Machine learning and data mining in pattern recognition*, pp. 35–49, 2003.
10. F. A. Thabtah, P. Cowling, and Y. Peng, "MMAC: A New Multi-class, Multi-label Associative Classification Approach," in *Proc. of the 4th IEEE International Conference on Data Mining*, pp. 217–224, 2004.
11. A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *Proc. of the 5th European Conference on PKDD*, pp. 42–53, 2001.
12. H. Blockeel, L. D. Raedt, and J. Ramon, "Top-down induction of clustering trees," in *Proc. of the 15th International Conference on Machine Learning*, pp. 55–63, 1998.
13. J. Fürnkranz, "Round robin classification," *Journal of Machine Learning Research*, vol. 2, pp. 721–747, 2002.
14. T.-F. Wu, C.-J. Lin, and R. C. Weng, "Probability estimates for multi-class classification by pairwise coupling," *Journal of Machine Learning Research*, vol. 5, pp. 975–1005, 2004.
15. X. Zhang, Q. Yuan, S. Zhao, W. Fan, W. Zheng, and Z. Wang, "Multi-label Classification without the Multi-label cost," in *Proc. of the 10th SIAM International Conference*

on Data Mining, 2010.
16. D. Kocev, *Ensembles for predicting structured outputs*. PhD thesis, IPS Jožef Stefan, Ljubljana, Slovenia, 2011.
17. J. Wang, Y. Zhao, X. Wu, and X.-S. Hua, "A transductive multi-label learning approach for video concept detection," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2274 – 2286, 2011.
18. J. Read, B. Pfahringer, and G. Holmes, "Multi-label Classification Using Ensembles of Pruned Sets," in *Proc. of the 8th IEEE International Conference on Data Mining*, pp. 995–1000, 2008.
19. J. Xu, "An extended one-versus-rest support vector machine for multi-label classification," *Neurocomputing*, vol. 74, no. 17, pp. 3114 – 3124, 2011.
20. G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-labelsets for multilabel classification," *IEEE Trans. on Knowl. and Data Eng.*, vol. 23, no. 7, pp. 1079–1089, 2011.
21. G. Tsoumakas, I. Katakis, and I. Vlahavas, "Effective and Efficient Multilabel Classification in Domains with Large Number of Labels," in *Proc. of the ECML/PKDD Workshop on Mining Multidimensional Data*, pp. 30–44, 2008.
22. J. Read, "A Pruned Problem Transformation Method for Multi-label classification," in *Proc. of the New Zealand Computer Science Research Student Conference*, pp. 143–150, 2008.
23. F. Tahi and H.-T. Lin, "Multi-Label Classification with Principle Label Space Transformation," in *Proc. of the 2nd International Workshop on Learning from Multi-Label Data*, pp. 45–52, 2010.
24. J. Fürnkranz, E. Hullermeier, E. L. Mencia, and K. Brinker, "Multi-label classification via calibrated label ranking," *Machine Learning*, vol. 73, no. 2, pp. 133–153, 2008.
25. S.-H. Park and J. Fürnkranz, "Efficient pairwise classification," in *Proc. of the 18th European Conference on Machine Learning*, pp. 658–665, 2007.
26. E. L. Mencía, S.-H. Park, and J. Fürnkranz, "Efficient voting prediction for pairwise multilabel classification," *Neurocomputing*, vol. 73, pp. 1164–1176, 2010.
27. G. Madjarov, D. Gjorgjevikj, and S. Džeroski, "Two stage architecture for multi-label learning," *Pattern Recognition*, vol. 45, no. 3, pp. 1019 – 1034, 2012.
28. J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning.*, vol. 85, no. 3, pp. 333–359, 2011.
29. P. Li, H. Li, and M. Wu, "Multi-label ensemble based on variable pairwise constraint projection," *Information Sciences*, vol. 222, no. 0, pp. 269 – 281, 2013.
30. X. Kong and P. S. Yu, "An ensemble-based approach to fast classification of multilabel data streams," in *Proc. of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2011.
31. R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.
32. A. Argyriou, T. Evgeniou, and M. Pontil, "Convex multi-task feature learning," *Machine Learning*, vol. 73, no. 3, pp. 243–272, 2008.
33. T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 109–117, 2004.
34. G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan, *Predicting Structured Data (Neural Information Processing)*. The MIT Press, 2007.
35. I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *International Conference on Machine Learning (ICML)*, pp. 104–112, 2004.

36. T. Joachims, T. Hofmann, Y. Yue, and C.-N. Yu, "Predicting structured objects with support vector machines," *Communications of the ACM, Research Highlight*, vol. 52, pp. 97–104, November 2009.
37. P. Geurts, L. Wehenkel, and F. d'Alché Buc, "Kernelizing the output of tree-based methods," in *Proc. of the 23rd international conference on Machine learning*, pp. 345–352, 2006.
38. C.-S. Ferng and H.-T. Lin, "Multi-label Classification with Error-correcting Codes," *Journal of Machine Learning Research - Proceedings Track*, vol. 20, pp. 281–295, 2011.
39. I. Katakis, G. Tsoumakas, and I. Vlahavas, "Multilabel Text Classification for Automated Tag Suggestion," in *Proc. of the ECML/PKDD Discovery Challenge*, 2008.
40. G. Nasierding, G. Tsoumakas, and A. Kouzani, "Clustering Based Multi-Label Classification for Image Annotation and Retrieval," in *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 4514–4519, 2009.
41. A. M. Kumar and M. Gopal, "A hybrid svm based decision tree," *Pattern Recognition*, vol. 43, pp. 3977–3987, 2010.
42. G. M. Dong and J. Chen, "Study on support vector machine based decision tree and application," in *Proc. of the 5th International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 318–322, 2008.
43. K. P. Bennett and J. A. Blue, "A Support Vector Machine Approach to Decision Trees," in *Proc. of the International Joint Conference on Neural Networks*, (Anchorage, Alaska), pp. 2396–2401, 1997.
44. K. Ting and L. Zhu, "Boosting support vector machines successfully," in *Multiple Classifier Systems*, pp. 509–518, 2009.
45. J. Gama, "Functional trees," *Machine Learning*, vol. 55, pp. 219–250, 2004.
46. J. R. Quinlan, *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, 1 ed., 1992.
47. G. Madjarov and D. Gjorgjevikj, "Hybrid decision tree architecture utilizing local svms for multi-label classification," in *Proceedings of the 7th international conference on Hybrid Artificial Intelligent Systems - Volume Part II*, HAIS'12, pp. 1–12, 2012.
48. C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.
49. L. Bottou and C.-J. Lin, "Support vector machine solvers," in *Large Scale Kernel Machines* (L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, eds.), pp. 301–320, MIT Press, 2007.
50. I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 1st ed., 2005.
51. G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data Mining and Knowledge Discovery Handbook*, pp. 667–685, Springer Berlin / Heidelberg, 2010.
52. M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.
53. P. Duygulu, K. Barnard, J. de Freitas, and D. Forsyth, "Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary," in *Proc. of the 7th European Conference on Computer Vision*, pp. 349–354, 2002.
54. A. Elisseeff and J. Weston, "A Kernel Method for Multi-Labelled Classification," in *Proc. of the Annual ACM Conference on Research and Development in Information Retrieval*, pp. 274–281, 2005.
55. B. Klimt and Y. Yang, "The Enron Corpus: A New Dataset for Email Classification Research," in *Proc. of the 15th European conference on Machine Learning*, pp. 217–226, 2004.
56. A. Srivastava and B. Zane-Ulman, "Discovering recurring anomalies in text reports

regarding complex space systems," in *Proc. of the IEEE Aerospace Conference*, pp. 55–63, 2005.
57. K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multilabel Classification of Music into Emotions," in *Proc. of the 9th International Conference on Music Information Retrieval*, pp. 320–330, 2008.
58. C. G. M. Snoek, M. Worring, J. C. van Gemert, J.-M. Geusebroek, and A. W. M. Smeulders, "The challenge problem for automated detection of 101 semantic concepts in multimedia," in *Proc. of the 14th Annual ACM International Conference on Multimedia*, pp. 421–430, 2006.
59. G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "Mulan: A java library for multi-label learning," *Journal of Machine Learning Research*, vol. 12, pp. 2411–2414, 2011.
60. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explorations*, vol. 11, pp. 10–18, 2009.
61. C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.