Journal Logo

00 (2011) 1–24

# Two Stage Architecture for Multi-label Learning

Gjorgji Madjarov[a,b], Dejan Gjorgjevikj[a], Sašo Džeroski[b]

[a] *Faculty of Electrical Engineering and Information Technology, Ss. Cyril and Methodius University, Skopje, Macedonia*
[b] *Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia*

## Abstract

A common approach to solving multi-label learning problems is to use problem transformation methods and dichotomizing classifiers as in the pair-wise decomposition strategy. One of the problems with this strategy is the need for querying a quadratic number of binary classifiers for making a prediction that can be quite time consuming, especially in learning problems with a large number of labels. To tackle this problem, we propose a Two Stage Architecture (TSA) for efficient multi-label learning. We analyze three implementations of this architecture the Two Stage Voting Method (TSVM), the Two Stage Classifier Chain Method (TSCCM) and the Two Stage Pruned Classifier Chain Method (TSPCCM). Eight different real-world datasets are used to evaluate the performance of the proposed methods. The performance of our approaches is compared with the performance of two algorithm adaptation methods (Multi-Label k-NN and Multi-Label C4.5) and five problem transformation methods (Binary Relevance, Classifier Chain, Calibrated Label Ranking with majority voting, the Quick Weighted method for pair-wise multi-label learning and the Label Powerset method). The results suggest that TSCCM and TSPCCM outperform the competing algorithms in terms of predictive accuracy, while TSVM has comparable predictive performance. In terms of testing speed, all three methods show better performance as compared to the pair-wise methods for multi-label learning.

*Keywords:*
multi-label learning, multi-label ranking, multi-label classification, two stage architecture, classifier chain

## 1. Introduction

The traditional problem of single-label classification is concerned with learning from examples, each associated with a single label $\lambda_i$ from a finite set of disjoint labels $L = \{\lambda_1, \lambda_2, ..., \lambda_Q\}$, $Q > 1$. For $Q > 2$, the learning problem is referred to as a *multi-class classification*. On the other hand, the task of learning a mapping from an example $x \in X$ ($X$ denotes the domain of examples) to a set of labels $Y \subseteq L$ is referred to as a *multi-label classification*. Thus, in contrast to multi-class classification, alternatives are not assumed to be mutually exclusive such that multiple labels may be associated with a single example i.e., each example can be a member of more than one class. The set of labels $Y$ are called relevant, while the set $L \backslash Y$ represents irrelevant labels 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 introduces the concept of *multi-label ranking* [1], which is understood as learning a model that the query example $x$ associates 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], 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 [2] and pair-wise method [11] [12] respectively. The computational complexity of the pair-wise learning approach to the multi-label scenario is large, especially in learning problems with a large number of labels.

In this paper, we propose a novel architecture for efficient pair-wise multi-label learning, named Two Stage Architecture (TSA). We analyze three different methods/implementations of this architecture the Two Stage Voting Method (TSVM), the Two Stage Classifier Chain Method (TSCCM) and the Two Stage Pruned Classifier Chain Method (TSPCCM). The two stage architecture and its three methods belong to the group of the problem transformation methods. Their main idea is to reduce the computational complexity of pair-wise methods and increase their predictive accuracy. We evaluate the performance of these methods on a selection of multi-label datasets that vary in terms of problem domain, number of labels and label cardinality. The obtained results demonstrate that our approaches outperform the competing methods (five problem transformation and two algorithm adaptation methods) in terms of predictive accuracy. Also, in terms of testing speed our architecture shows better performance as compared to the pair-wise methods for multi-label learning.

For the readers' convenience, Section 2 surveys some previous work in multi-label learning. The Two Stage Architecture and its computational complexity are presented in Section 3. Section 4 presents the experimental results, that compare the performance of the proposed approaches (TSVM, TSCCM and TSPCCM) with other competing methods. The conclusion and directions for further work are given in Section 5.

## 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 two main categories: Algorithm adaptation methods and problem transformation 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

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.

Clare et al. [13] adapted the C4.5 algorithm for multi-label data (ML-C4.5). They modified the formula of entropy calculation (equation 1) in order to solve the multi-label problem. They also allowed multiple labels in the leaves of the tree. The modified entropy sums the entropies for each individual class label.

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

where $S$ is the set of examples, $p(c_i)$ is the relative frequency of class $c_i$ and $q(c_i) = 1 - p(c_i)$.

ML-kNN [3] is based on the popular k Nearest Neighbors (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 neighbors. Other kNN based approaches for multi-label learning also exist [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*

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 labeled 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. A method closely related to the BR method is the Classifier Chain method (CC) proposed by Read et al. [14]. 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.

Second problem transformation method is the label combination method, or label power-set method, (LP), which has been the focus of several recent studies [15] [16] [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 $(x, Y)$ 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 [17] developed a pruned problem transformation method (PPT), that selects only the transformed labels that occur more than predefined number of times. A disadvantage of these methods, however, is their worst-case time complexity.

Third problem transformation approach to solving the multi-label learning problem by using binary classifiers is pair-wise classification or round robin classification [11] [12]. Its basic idea is to use $Q*(Q-1)/2$ classifiers covering all pairs of labels. Each classifier is trained using the samples of the first label as positive examples and the samples of the second label as negative examples. 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 only the relevant labels for each example a label ranking algorithm is used.

Brinker et al. [1] 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 $\lambda_0$, which will represent the split-point between relevant and irrelevant labels. The calibration label $\lambda_0$ 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 $\lambda_0$). CLR is considered a combination of multi-label classification and ranking.

Besides majority voting in CLR, Park et al. [18] 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. It exploits the fact that during voting some classes can be excluded from the set of possible top rank classes early in the process, when it becomes clear that even if they reach the maximal voting mass in the remaining evaluations they can not exceed the current maximum. Pairwise classifiers are selected depending on a voting loss value, which is the number of votes that a class has not received. The voting loss starts with a value of zero and increases monotonically with the number of performed preference evaluations. The class with the current minimal loss is the best candidate for the top ranked class. If all preferences involving this class have been evaluated (and it still has the lowest loss), it can be concluded that no other class can achieve a better ranking. Thus, the QWeighted algorithm always focuses on classes with low voting loss. An adaptation of QWeighted to multi-label learning (QWeightedML) [19] is to repeat the process while all relevant labels are not determined, i.e., until the returned label is the artificial label $\lambda_0$, which means that all remaining labels will be considered to be irrelevant.

Several ensemble approaches have been developed based on the common problem transformation methods. Good examples are the RAKEL system by Tsoumakas et al. [15] and ensembles of classifier chains (ECC) [14]. Note that binary methods are occasionally referred to as ensemble methods because they involve multiple binary models. 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.

The Calibrated Label Ranking method (CLR) often demonstrates better prediction accuracy as compared to the other non-ensemble methods for multi-label learning. Its major problem is the need for querying a quadratic number of classifiers for making a single prediction. That can be very time consuming especially in learning problems with a large number of labels.

## 3. Two Stage Architecture (TSA) and its implementations

### 3.1. Two Stage Architecture (TSA)

In this paper, we propose a novel Two Stage Architecture (TSA) for efficient pair-wise multi-label learning that is related to the CLR algorithm [20]. The main idea of this architecture is to reduce the number of classifiers that are needed to be consulted in the prediction phase of the CLR algorithm and increase the predictive accuracy. We first introduce the architecture and then present three different methods/implementations of this architecture: the Two Stage Voting Method (TSVM), the Two Stage Classifier Chain Method (TSCCM) and the Two Stage Pruned Classifier Chain Method (TSPCCM). This section concludes with an analysis of the computational complexity of these methods.

The conventional pair-wise approach learns a model $M_{ij}$ for all combinations of labels $\lambda_i$ and $\lambda_j$, $1 \leq i < j \leq Q$. In this way $Q * (Q-1)/2$ different pair-wise models are learned. Each pair-wise model $M_{ij}$ is a traditional one-against-one binary classifier and is learned with the $\lambda_i$ examples labeled as positive and the $\lambda_j$ examples labeled as negative. This means that, for a given training set $S = \left\{(x_1, Y_1), (x_2, Y_2), ..., (x_p, Y_p)\right\}$ ($x_i \in X, Y_i \subseteq L$, where $X$ denotes the domain of examples and $L = \{\lambda_1, \lambda_2, ..., \lambda_Q\}$ is a finite set of labels), each model $M_{ij}$ is trained with the examples $(x_r, Y'_r)$ $(0 < r \leq p)$ where $Y'_r$ is defined as:

$$Y'_r = \begin{cases} +1, & \text{if } \lambda_i \in Y_r \text{ and } \lambda_j \notin Y_r & \text{(2a)} \\ -1, & \text{if } \lambda_j \in Y_r \text{ and } \lambda_i \notin Y_r & \text{(2b)} \end{cases}$$

This transformation of the dataset is known as the two-label transformation. If the training example $x_r$ is labeled with the labels $\lambda_i$ and $\lambda_j$ at the same time, it is not involved in the learning process of the model $M_{ij}$. It cannot be used to distinguish between labels $\lambda_i$ and $\lambda_j$ and cannot be viewed as either a positive or a negative example. In the prediction process the other pair-wise models $M_{il}$ ($i < l \leq Q$ and $l \neq j$) and $M_{li}$ ($1 \leq l < i$ and $l \neq j$) for label $\lambda_i$ and $M_{jl}$ ($j < l \leq Q$ and $l \neq i$) and $M_{lj}$ ($1 \leq l < j$ and $l \neq i$) for label $\lambda_j$ are expected to make the decision on whether these labels are relevant or not. The model $M_{ij}$ only votes about which label ($\lambda_i$ or $\lambda_j$) should be better ranked. The main disadvantage of this approach is that in the prediction process a quadratic number of base classifiers (models) have to be consulted for each test example.

As a result of introducing the artificial calibration label $\lambda_0$ in the calibrated label ranking algorithm [20], the number of the base classifiers is increased by $Q$ i.e., an additional set of $Q$ binary preference models $M_{k0}$ ($1 \leq k \leq Q$) is learned. The models $M_{k0}$ that are learned by a pair-wise approach to calibrated ranking, and the models $M_k$ that are learned by conventional binary relevance are equivalent. By definition, for the binary relevance (traditional one-against-all) approach, the training example $x$ is a positive example in the training set for the model $M_k$, if the label $\lambda_k$ is a relevant label for the training example $x$. Similarly, if the label $\lambda_k$ is irrelevant for the training example $x$, $x$ is negative example for learning the model $M_k$. Using the same notation as in the case of the pair-wise models ($M_{ij}$), each model $M_{k0}$ is trained with the examples $(x_r, Y'_r)$ ($r \in 1...p$) where $Y'_r$ is defined as:

$$Y'_r = \begin{cases} +1, & \text{if } \lambda_k \in Y_r & \text{(3a)} \\ -1, & \text{if } \lambda_k \notin Y_r & \text{(3b)} \end{cases}$$

This transformation of the datasets is addressed as single-label transformation.

The binary relevance models $M_k$ $(1 \leq k \leq Q)$ almost always have higher time complexity than pair-wise models $M_{ij}$ $(1 \leq i < j \leq Q)$ because they are learned with all the examples from the training set, while the pair-wise models $M_{ij}$ are learned only with the examples labeled with labels $\lambda_i$ and $\lambda_j$. In the standard voting algorithm for calibrated label ranking, each test example needs to consult all the models (classifiers) $M_k$ $(1 \leq k \leq Q)$ and $M_{ij}$ $(1 \leq i < j \leq Q)$ in order to rank the labels by their order of preference.

As a result of the increased number of models, the CLR method leads to more accurate prediction, but also leads to slower testing and higher computational complexity, especially when the number of the labels in the problem is large.

The Two Stage Architecture is organized in two layers (Figure 1). The first layer has $Q$ binary relevance models $M_{k0}$, while the second layer has $Q * (Q-1)/2$ pair-wise models $M_{ij}$. Each model $M_{k0}$ from the first layer is connected to $Q-1$ models $M_{ij}$ from the second layer, where $k = i$ or $k = j$ $(1 \leq i \leq Q-1, i+1 \leq j \leq Q)$. It is responsible for learning and predicting the probability association of label $\lambda_k$. On the other hand, each model of the second layer $M_{ij}$ is connected to exactly two binary relevance models from the first layer ($M_{i0}$ and $M_{j0}$) and is responsible for learning and predicting the probability associations of label $\lambda_i$ and label $\lambda_j$ $(p(\lambda_i) = 1 - p(\lambda_j))$.

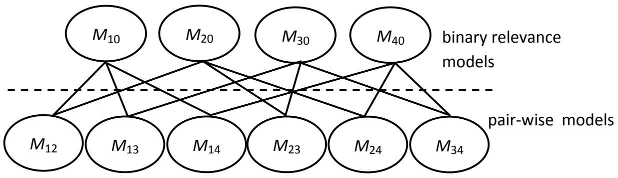

Figure 1: Two Stage Architecture.

In the following subsections, we present three different implementations of this architecture (Two Stage Voting Method - TSVM, Two Stage Classifier Chain Method - TSCCM and Two Stage Pruned Classifier Chain Method - TSPCCM). All of them have the same organization of the binary relevance and the pair-wise models in the two layers. Each method itself introduces a different approach for training and testing the classifiers from the architecture. All three methods improve the testing speed in comparison to the pair-wise methods. TSCCM and TSPCCM also improve the predictive accuracy, as a result of the classifier chain which will be explained in the next subsection, while the TSVM has comparable predictive performance to the competing methods.

### 3.2. Two Stage Voting Method - TSVM

The training phase of TSVM is the same as in the CLR method. Each model of the architecture is trained with the corresponding examples of a given training dataset. The training procedure of the first and the second layer is outlined in Table 1. Recall the notation for a training example $(x, Y)$, where $Y \subseteq L$ $(L = \{\lambda_1, \lambda_2, ..., \lambda_Q\})$, and $x$ is an instance feature vector.

Table 1: The training procedure of the TSVM.

```
TRAINING FIRST LAYER - TSVM
($S = \{(x_1, Y_1), ..., (x_p, Y_p)\}$)
for $k \in 1, ..., Q$ do
    $S_{k0} = SingleLabelTransformation(S, \lambda_k)$
    $M_{k0} = TrainingModel(S_{k0})$
TRAINING SECOND LAYER - TSVM
($S = \{(x_1, Y_1), ..., (x_p, Y_p)\}$)
for $i \in 1, ..., Q-1$ do
    for $j \in i+1, ..., Q$ do
        $S_{ij} = TwoLabelTransformation(S, \lambda_i, \lambda_j)$
        $M_{ij} = TrainingModel(S_{ij})$
```

In the prediction phase, each model $M_{k0}$ tries to determine the relevant labels for the corresponding test example. Each model $M_{k0}$ gives the probability (the output value of model $M_{k0}$ is converted to probability) that the test example is associated with the label $\lambda_k$. If that probability is appropriately small (under some predetermined threshold), we can conclude that the artificial calibration label $\lambda_0$ is preferred over the label $\lambda_k$, i.e., the label $\lambda_k$ belongs to the set of irrelevant labels. In that case, we conclude that, the pair-wise models of the second layer $M_{ij}$ where $i = k$ or $j = k$, do not need to be consulted for the corresponding test example, because the binary relevance model $M_{k0}$ from the first layer has suggested that the label $\lambda_k$ belongs to the set of irrelevant labels. For each label $\lambda_k$ that belongs to the set of irrelevant labels, the number of pair-wise models that should be consulted decreases of $Q - 1$.

In order to decide which labels belong to the set of irrelevant labels i.e., which pair-wise models $M_{ij}$ from the second layer do not have to be consulted, a threshold $t$ $(0 \leq t \leq 1)$ is introduced.

As previously described, every test example first consults all binary relevance models $M_{k0}$ of the first layer of the two stage architecture. For each corresponding model $M_{k0}$ $(1 \leq k \leq Q)$ its output is converted to probability and compared to the threshold $t$.

- If the prediction probability is above the threshold, the test example is forwarded to all the models $M_{ij}$ of the second layer of the architecture that are associated to the model $M_{k0}$.

- If the prediction probability is under the threshold, the test example is not forwarded from $M_{k0}$ to any model of the second layer of the architecture.

From the viewpoint of the pair-wise models $M_{ij}$, if we consider the prediction probabilities of the binary relevance models $M_{i0}$ and $M_{j0}$ of the first layer, three distinct cases in the voting process can appear:

1. The prediction probabilities of both binary relevance models $M_{i0}$ and $M_{j0}$ that are connected to the pair-wise model $M_{ij}$ are above the threshold $t$.
2. The prediction probability of only one of the binary relevance models ($M_{i0}$ or $M_{j0}$) is above the threshold $t$.
3. The prediction probabilities of the binary relevance models $M_{i0}$ and $M_{j0}$ are both under the threshold $t$.

In the first case, the model $M_{ij}$ is consulted and its prediction is decoded into a vote for one of the labels $\lambda_i$ or $\lambda_j$. In the second case, $M_{ij}$ is not consulted and its vote goes directly to the label whose binary relevance model prediction probability is above the threshold $t$. In the third case $M_{ij}$ is not consulted and it does not vote at all. The votes of all $M_{k0}$ models and $M_{ij}$ models (where at least one prediction probability of the models $M_{i0}$ and $M_{j0}$ is above the threshold $t$) are then aggregated to obtain the final prediction by majority voting. The classification process is outlined in Table 2.

By increasing the value of the threshold, the number of pair-wise models that should be consulted decreases. For $t = 1$ no example is forwarded to the second layer of the architecture and the decision is made by the classifiers of the first layer. On the other hand, for $t = 0$, for each test example all pair-wise models of the second layer are consulted and TSVM becomes CLR.

### 3.3. Two Stage Classifier Chain Method - TSCCM

In this subsection we propose a modification of the Two Stage Voting Method (TSVM) that uses a classifier chain in order to improve the predictive accuracy. In this context, the term classifier chain can be explained as extending the feature space of each example involved in the learning and in the testing process of the model $M_{ij}$ from the second layer with the probability predictions of all binary relevance models $M_{k0}$ (the models from the first layer) for that example. This means that the feature space of each link which connects the binary relevance models and the pair-wise models is extended with the probability predictions of all models of the first layer. This approach will be referred to as Two Stage Classifier Chain Method (TSCCM).

All models ($M_{k0}$) in the first layer of the architecture in TSCCM are learned in the same way as in TSVM. The only difference between TSVM and TSCCM is in the learning phase of the models from the second layer of the architecture. In this phase, all training examples consult the models $M_{k0}$ first, and then the feature space of each example is extended with the probability predictions of those models. The training examples with extended feature vectors are used in the learning phase of the models from the second layer of the architecture in TSCCM. In the

Table 2: Testing procedure of the TSVM.

```
CLASSIFY($x$) - TSVM
for $k \in 1, ..., Q$ do
    $probability(M_{k0}) = classify(M_{k0}(x))$
    if $probability(M_{k0}) > 0.5$ then
        votes[$\lambda_k$]++
    else
        votes[$\lambda_0$]++
for $i \in 1, ..., Q - 1$ do
    for $j \in i + 1, ..., Q$ do
        if $probability(M_{i0}) > t$ & $probability(M_{j0}) > t$ then
            $//first\ case$
            $probability(M_{ij}) = classify(M_{ij}(x))$
            if $probability(M_{ij}) > 0.5$ then
                votes[$\lambda_i$]++
            else
                votes[$\lambda_j$]++
        else
            $//second\ case$
            if $probability(M_{i0}) > t$ then
                votes[$\lambda_i$]++
            if $probability(M_{j0}) > t$ then
                votes[$\lambda_j$]++
$order$(votes[ ])
```

testing phase, before an example is forwarded to the models in the second layer, its feature space is extended with the probability predictions of all models from the first layer.

TSCCM reduces the number of base classifiers that need to be consulted in order to make a final prediction for a given test example in the same way as TSVM. Besides, it also improves the predictive performance as a result of the introduced classifier chain. The training procedure of TSCCM is outlined in Table 3, while the classification process is outlined in Table 4. We use the same notation as for TSVM.

### 3.4. Two Stage Pruned Classifier Chain Method - TSPCCM

As mentioned above, the feature space of each link which connects the binary relevance and the pair-wise models in TSCCM is extended with the probability predictions of all binary relevance models (all models of the first layer). In this subsection we propose a modification of the TSCCM in which the feature space of each example, involved in the learning and testing process of the model $M_{ij}$ of the second layer, is extended not by all but only with the probability predictions of the binary relevance models $M_{i0}$ and $M_{j0}$, i.e., the models that are directly connected to the model $M_{ij}$. These predictions are the most relevant for the model $M_{ij}$ because the model $M_{ij}$ tries to distinguish the label $\lambda_i$ from the label $\lambda_j$. This approach will be called the Two Stage Pruned Classifier Chain Method (TSPCCM). The training procedure of TSPCCM is outlined in Table 5, while the classification process is outlined in Table 6. We use the same notation as for TSVM and for TSCCM.

### 3.5. Computational Complexity

The computational complexities of the calibrated label ranking method and the two stage voting method (TSVM) in the learning phase are equal, because the models learned by both methods are practically the same. On the other hand, in spite of the equal number of learned models, the computational complexity of the two stage classifier chain and the two stage pruned classifier chain methods (TSCCM and TSPCCM) in the learning phase is slightly higher than CLR, as a result of the increase in the feature vector size in each model of the second layer of the architecture.

Table 3: Training procedure of the TSCCM.

```
TRAINING FIRST LAYER - TSCCM
($S = \{(x_1, Y_1), ..., (x_p, Y_p)\}$)
for $k \in 1, ..., Q$ do
    $S_{k0} = SingleLabelTransformation(S, \lambda_k)$
    $M_{k0} = TrainingModel(S_{k0})$
$S' \leftarrow \{\}$
for $(x, Y) \in S$ do
    $x' \leftarrow x$
    for $k \in 1, ..., Q$ do
        $probability(M_{k0}) = classify(M_{k0}(x))$
        $x' \leftarrow x' \cup (probability(M_{k0}))$
    $S' \leftarrow S' \cup (x', Y)$
TRAINING SECOND LAYER - TSCCM
($S' = \{(x'_1, Y_1), ..., (x'_p, Y_p)\}$)
for $i \in 1, ..., Q - 1$ do
    for $j \in i + 1, ..., Q$ do
        $S'_{ij} = TwoLabelTransformation(S', \lambda_i, \lambda_j)$
        $M_{ij} = TrainingModel(S'_{ij})$
```

Table 4: Testing procedure of the TSCCM.

```
CLASSIFY($x$) - TSCCM
$x' \leftarrow x$
for $k \in 1, ..., Q$ do
    $probability(M_{k0}) = classify(M_{k0}(x))$
    if $probability(M_{k0}) > 0.5$ then
        votes[$\lambda_k$]++
    else
        votes[$\lambda_0$]++
    $x' \leftarrow x' \cup (probability(M_{k0}))$
for $i \in 1, ..., Q - 1$ do
    for $j \in i + 1, ..., Q$ do
        if $probability(M_{i0}) > t$ & $probability(M_{j0}) > t$ then
            $//first\ case$
            $probability(M_{ij}) = classify(M_{ij}(x'))$
            if $probability(M_{ij}) > 0.5$ then
                votes[$\lambda_i$]++
            else
                votes[$\lambda_j$]++
        else
            $//second\ case$
            if $probability(M_{i0}) > t$ then
                votes[$\lambda_i$]++
            if $probability(M_{j0}) > t$ then
                votes[$\lambda_j$]++
order(votes[ ])
```

Table 5: Training procedure of the TSPCCM.

```
TRAINING FIRST LAYER - TSPCCM
($S = \{(x_1, Y_1), ..., (x_p, Y_p)\}$)
for $k \in 1, ..., Q$ do
    $S_{k0} = SingleLabelTransformation(S, \lambda_k)$
    $M_{k0} = TrainingModel(S_{k0})$

TRAINING SECOND LAYER - TSPCCM
($S = \{(x_1, Y_1), ..., (x_p, Y_p)\}$)
for $i \in 1, ..., Q - 1$ do
    for $j \in i + 1, ..., Q$ do
        $S' \leftarrow \{\}$
        for $(x, Y) \in S$ do
            $x' \leftarrow x$
            $probability(M_{i0}) = classify(M_{i0}(x))$
            $x' \leftarrow x' \cup (probability(M_{i0}))$
            $probability(M_{j0}) = classify(M_{j0}(x))$
            $x' \leftarrow x' \cup (probability(M_{j0}))$
            $S' \leftarrow S' \cup (x', Y)$
        $S'_{ij} = TwoLabelTransformation(S', \lambda_i, \lambda_j)$
        $M_{ij} = TrainingModel(S'_{ij})$
```

The increase of the computational complexity in the learning phase of TSCCM as compared to CLR, depends on the number of labels in the multi-label learning problem. If this number is small, as compared to the number of features in the learning problem, then the computational complexity of the learning phase of TSCCM is similar to the computational complexity of the learning phase of CLR. In most classification problems, the number of features is significantly larger (more than 10 times) than the number of labels, so the training times of TSCCM are inconsiderably longer than those of CLR. The computational complexity in the learning phase of TSPCCM is practically the same as that of CLR, because in this case the feature vectors of the training examples are extended by only two additional features for each model located in the second layer of the architecture. However, the computational complexity of the training phase of the pair-wise approaches for problems with a large number of labels, remains the main drawback of the methods proposed here as well, as a result of the quadratic number of models that should be learned.

In the rest of this section, we analyze the computational complexity of the TSA comparing to the computational complexity of the CLR method in the prediction phase. Since the main idea of the proposed architecture is to reduce the number of consulted models in the prediction process, the biggest advance is expected here. In the following, we use the term computational complexity, strictly in the sense of a computational complexity in the prediction phase.

The computational complexity of TSA significantly differs from the computational complexity of CLR. The computational complexity of the calibrated label ranking method ($O_{CLR}$) can be defined as the sum of the computational complexity of the binary relevance models ($O_{BR}$) and the pair-wise models ($O_P$):

$$O_{CLR} = O_{BR} + O_P \tag{4}$$

The computational complexity of TSA can be defined as the sum of the computational complexity of the models located in the first layer of the architecture ($O_{FL}$) and the computational complexity of the models located in the second layer of the architecture ($O_{SL}$):

$$O_{TSA} = O_{FL} + O_{SL} \tag{5}$$

The computational complexity of the first layer of the TSA and the computational complexity of the binary relevance models of the CLR method are equal ($O_{BR} = O_{FL}$). In both methods, the models are the same and each test example must consult all of these models in order to predict the class the example belongs to.

Table 6: Testing procedure of the TSPCCM.

```
CLASSIFY($x$) - TSPCCM
for $k \in 1, ..., Q$ do
    $probability(M_{k0}) = classify(M_{k0}(x))$
    if $probability(M_{k0}) > 0.5$ then
        votes[$\lambda_k$]++
    else
        votes[$\lambda_0$]++
for $i \in 1, ..., Q - 1$ do
    for $j \in i + 1, ..., Q$ do
        if $probability(M_{i0}) > t$ & $probability(M_{j0}) > t$ then
            $//first\ case$
            $x' \leftarrow x$
            $x' \leftarrow x' \cup (probability(M_{i0}))$
            $x' \leftarrow x' \cup (probability(M_{j0}))$
            $probability(M_{ij}) = classify(M_{ij}(x'))$
            if $probability(M_{ij}) > 0.5$ then
                votes[$\lambda_i$]++
            else
                votes[$\lambda_j$]++
        else
            $//second\ case$
            if $probability(M_{i0}) > t$ then
                votes[$\lambda_i$]++
            if $probability(M_{j0}) > t$ then
                votes[$\lambda_j$]++
$order$(votes[ ])
```

The main difference in computational complexity between CLR and TSA (TSVM, TSCCM and TSPCCM) is in the computational complexity of the pair-wise models of CLR and the second layer of TSA. As noted in the previous section, if the threshold is set to zero ($t = 0$), in the TSA method, all models of the second layer are consulted and we have $O_{SL} = O_P$ for TSVM. For TSCCM and TSPCCM this equality is only approximal $O_{SL} \approx O_P$ as a result of the extended feature vector that adds some complexity to the classifiers themselves. If the threshold is set to one ($t = 1$), no models of the second layer will be consulted, so $O_{SL}$ will be 0 and $O_{TSA} = O_{FL} = O_{BR}$. For threshold values $0 < t < 1$, $O_{SL} = r * O_P$ where $r$ is a reduction parameter specific for each multi-label dataset ($0 < r < 1$). The reduction parameter $r$ is related to label cardinality ($l_c$) [2], i.e., the average number of relevant labels per example in a given multi-label dataset.

As mentioned in the previous section, if the binary relevance model of the first layer of TSA predicts a value that is above the threshold $t$, the test example will be forwarded to the second layer. If all of the models from the first layer of TSA forward the test example to the second layer, all pair-wise models from the second layer will be consulted and the number of consulted pair-wise models becomes $Q * (Q - 1)/2$ (TSVM becomes CLR). If only one model $M_{k0}$ ($1 \leq k \leq Q$) from the first layer of TSA forwards the test example to the second layer, it means that only the label $\lambda_k$ is preferred over the calibration label $\lambda_0$ and all the votes from the pair-wise models $M_{ik}$ and $M_{kj}$ ($1 \leq i < k, k < j \leq Q$) go to $\lambda_k$. In this case, the pair-wise models $M_{ik}$ and $M_{kj}$ are not consulted and the $O_{SL}$ is still equal to 0. If two models $M_{m0}$ and $M_{n0}$ ($1 \leq m \leq Q, 1 \leq n \leq Q, m \neq n$) from the first layer of TSA forward the test example to the second layer, only one pair-wise model $M_{mn}$ ($m < n$) or $M_{nm}$ ($n < m$) is consulted and the label that will win the vote from this model will be the top ranked label, while the other label of the model will be ranked second. If we assume nearly ideal prediction by the binary relevance models from the first layer of the TSA, the number of pair-wise models consulted for one test example becomes $rl * (rl - 1)/2$, where $rl$ is the number of relevant labels, i.e., the number of labels that are preferred over the calibration label $\lambda_0$ for the corresponding test example. This means that, the average number of pair-wise models consulted for each test example from the dataset, becomes $l_c * (l_c - 1)/2$, where $l_c$ is the label cardinality of the multi-label dataset mentioned above. In this (ideal) case (prediction accuracy of 100% by the binary relevance models), the reduction parameter $r$ can be determined as:

$$r = \frac{l_c * (l_c - 1)}{Q * (Q - 1)} \tag{6}$$

However, for a real world problem the reduction parameter $r_{real}$ will be:

$$r_{real} = \frac{a_{brmf} * (a_{brmf} - 1)}{Q * (Q - 1)} \tag{7}$$

where $a_{brmf}$ is the average number of binary relevance models located in the first layer of TSA that give a probability that is above the threshold $t$ in the prediction process. The value of the parameter $r_{real}$ is always greater than or equal to the value of the parameter $r$ ($r_{real} \geq r$).

## 4. Experiments

In this section, we present the results of our experiments with several algorithms on a number of multi-label classification problems. The problems come from the areas of classification of text, music, images and gene function. We compare the performance of the proposed methods (TSVM, TSCCM and TSPCCM) to the performance of two algorithm adaptation methods (Multi-Label k-NN - ML-kNN [3] and Multi-Label C4.5 - ML-C4.5 [13]) and five problem transformation methods (Binary Relevance - BR [2], Classifier Chain - CC [14], Calibrated Label Ranking with majority voting - CLR [20], the Quick Weighted method for pair-wise multi-label classification - QWeightedML [19] and the Label Powerset - LP method [2]).

### *4.1. Evaluation metrics*

For a given training set $S = \left\{(x_1, Y_1), (x_2, Y_2), ..., (x_p, Y_p)\right\}$ ($x_i \in X, Y_i \subseteq L$) where $X$ denote the domain of examples and $L = \{\lambda_1, \lambda_2, ..., \lambda_Q\}$ is a finite set of labels, the goal of the learning system is to output a multi-label classifier $h : X \rightarrow 2^L$. In most cases, the learning system will also produce a real-valued function of the form $f : X \times L \rightarrow R$. It is supposed that, for a given example $(x_i, Y_i)$, a successful learning system will tend to output larger values for labels in $Y_i$ than those not in $Y_i$, i.e. $f(x_i, \lambda_m) > f(x_i, \lambda_n)$ for any $\lambda_m \in Y_i$ and $\lambda_n \notin Y_i$.

Performance evaluation for multi-label learning systems thus differs from that of classical single-label learning systems. Four different multi-label evaluation metrics (Hamming Loss, One Error, Coverage and Average Precision) proposed in [8] are used in this paper. The first one, Hamming Loss is defined in terms of the function $h(x)$ that predicts the set of labels for a given example $x$. The definitions of the other three metrics are based on the real-valued function $f(\bullet, \bullet)$ that takes into account the ranking quality of different labels for each example. One Error is a simple generalization of classification error for multi-label problems, while the other two evaluation metrics (Coverage and Average Precision) are based on measures used in information retrieval and are used to evaluate the performance of the various learning algorithms in terms of their label rankings. We summarize their definitions below.

1. Hamming loss evaluates how many times an example-label pair is misclassified, i.e., throughout a label not belonging to the instance is predicted or a label belonging to the instance 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:

$$hamming\_loss(h) = \frac{1}{p} \sum_{i=1}^{p} \frac{1}{Q} |h(x_i) \Delta Y_i| \tag{8}$$

where $\Delta$ stands for the symmetric difference between two sets and $Q$ is the total number of possible class labels.

2. 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:

$$one\_error(f) = \frac{1}{p}\sum_{i=1}^{p}\left[\!\left[\left[\arg\max_{\lambda\in Y} f(x_i,\lambda)\right] \notin Y_i\right]\!\right] \tag{9}$$

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

3. Coverage evaluates how far, on average we need to go down the list of ranked labels in order to cover all the relevant labels of the instance. The smaller the value of $coverage(f)$, the better the performance.

$$coverage(f) = \frac{1}{p}\sum_{i=1}^{p}\max_{\lambda\in Y_i} rank_f(x_i,\lambda) - 1 \tag{10}$$

   where $rank_f(x_i,\lambda)$ maps the outputs of $f(x_i,\lambda)$ for any $\lambda \in L$ to $\{\lambda_1, \lambda_2, ..., \lambda_Q\}$ so that $f(x_i,\lambda_m) > f(x_i,\lambda_n)$ implies $rank_f(x_i,\lambda_m) < rank_f(x_i,\lambda_n)$. The smallest possible value for $coverage(f)$ is $l_c$, i.e., the label cardinality of the given dataset.

4. Average Precision is the average fraction of labels ranked above an actual label $\lambda \in Y_i$ that actually are in $Y_i$. The performance is perfect when $avg\_precision(f) = 1$; the larger the value of $avg\_precision(f)$, the better the performance. This metric is defined as:

$$avg\_precision(f) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{|Y_i|}\sum_{\lambda\in Y_i}\frac{|L_i|}{rank_f(x_i,\lambda)} \tag{11}$$

   where $L_i = \{\lambda' | rank_f(x_i,\lambda') \le rank_f(x_i,\lambda), \lambda' \in Y_i\}$ and $rank_f(x_i,\lambda)$ is defined as in coverage above.

### *4.2. Experimental questions*

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

1. Does the Two Stage Architecture improve the predictive performance and computational complexity in the prediction phase over the CLR method?
2. Are the TSCCM and the TSPCCM better than the TSVM as a result of the additional information in the feature vector in each model of the second layer of the architecture?
3. Is the TSPCCM more accurate and efficient in the prediction phase than TSCCM?
4. Is there a dependence between the answers of the previous three questions and the properties of the datasets (such as number of labels and label cardinality)?
5. What is the dependence between the predictive performance and the testing time of the proposed methods and the values of the threshold $t$?
6. What are the predictive performance and testing times of the ten different methods?

In order to answer these questions, we evaluate the predictive performance and the computational complexity in the prediction phase of the ten methods. The evaluation is conducted on a selection of multi-label datasets that vary in terms of problem domain, number of labels and label cardinality (question 4). To address questions 1, 2 and 3, we have employed the non-parametric Wilcoxon test for statistical significance [21] and we discuss in detail the results obtained by the proposed methods and CLR.

To access the dependence of the predictive performance and the computational complexity in the prediction phase on different values of the threshold $t$ (question 5), the performance in terms of the four metrics defined above and testing times were recorded for the threshold values 0.0 to 1.0 with step 0.1, for each proposed method. To answer the question 6, we compare the performance of the proposed and competing methods in terms of predictive accuracy and testing speed. The corrected Friedman test [22] and the post hoc Nemenyi test [23] were employed to assess whether the differences in performance between the different approaches are statistically significant.

### 4.3. Datasets and experimental setup

In our experiments, eight different multi-label classification problems were addressed by each of the mentioned multi-label learning methods. The predictive performance in terms of the metrics defined above and testing times were recorded for every method for each classification problem. The problems considered in the experiments include:

1. image classification: scene [24] and corel5k [25];
2. gene function classification: yeast [26];
3. text classification: enron [27], medical [14], bibtex [28] and tmc2007 [29];
4. music classification: emotions [30];

The complete description of the datasets in terms of the number of training (#*tr.e.*) and test (#*t.e.*) examples, the number of features (#*f.*), the total number of labels (#*l.*) and label cardinality ($l_c$) are shown in Table 7.

Table 7: Dataset description.

| | #*tr.e.* | #*t.e.* | #*f.* | #*l.* | $l_c$ |
|---|---|---|---|---|---|
| **scene** | 1211 | 1159 | 294 | 6 | 1.07 |
| **yeast** | 1500 | 917 | 103 | 14 | 4.24 |
| **enron** | 1123 | 579 | 1001 | 53 | 3.38 |
| **emotions** | 391 | 202 | 72 | 6 | 1.87 |
| **tmc2007** | 21519 | 7077 | 49060 | 22 | 2.16 |
| **medical** | 645 | 333 | 1449 | 45 | 1.25 |
| **bibtex** | 4880 | 2515 | 1836 | 159 | 2.40 |
| **corel5k** | 4500 | 500 | 499 | 374 | 3.52 |

The training and testing of the proposed methods were performed using a custom developed application that uses the MULAN library[1] for the machine learning framework Weka [31]. We also implemented the ML-C4.5 and CC methods under the same library. The other considered methods are already implemented in MULAN.

The LIBSVM library [32], and in particular SVM 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 the datasets were determined by 5-fold cross validation using only the training sets. The outputs of the SVM classifiers are converted to probabilities using the LIBSVM implementation for extending SVM with probability estimates [32]. In all classification problems the classifiers were trained using all available training examples and were evaluated by recognizing all test examples from the corresponding dataset.

Table 8 shows the values of the threshold $t$ for each dataset separately, for which the presented results of the proposed methods are obtained. The value of the threshold $t$ for each dataset for TSVM, TSCCM and TSPCCM was determined by 5-fold cross validation on the training set. This was done in order to achieve optimal performance (trade off) in terms of the four evaluation metrics and the computational efficiency. The values 0.005 to 0.01 with step 0.001, 0.01 to 0.1 with step 0.01 and 0.1 to 1.0 with step 0.05 were considered for $t$.

### 4.4. Results

Tables 9 to 13 give the performance of each method on each of the datasets measured in terms of the four performance metrics and testing speed. The first column of the tables lists the method, while the remaining columns show the performance of each method for every dataset. Tables 9 to 12 show the predictive performance in terms of Hamming Loss, One Error, Coverage and Average Precision, irrespectively. The best results per dataset in these tables are shown in boldface. The testing time of each method on each of the datasets measured in seconds, are given in Table 13. The testing times of the algorithm adaptation methods (ML-kNN and ML-C4.5), binary relevance methods (BR and CC), label power-set method (LP) and the pair-wise methods (CLR, QWeightedML, TSVM, TSCCM and

[1] http://mulan.sourceforge.net/

TSPCCM) are visually separated. This is done in order to make an easier and better comparison between the results obtained by the methods from these four groups and the methods from each group separately (such as the group of the pair-wise methods that attracts the most attention).

The results in Tables 9 to 12 clearly show that among the ten tested approaches, TSCCM and TSPCCM offer better predictive performance than the other algorithms, in almost every evaluation metric for all multi-label classification problems. TSPCCM is significantly better (according to the non-parametric Wilcoxon test for statistical significance) than CLR (with a significance level $p < 0.018$) in terms of One Error and ($p < 0.012$) in terms of testing time. TSCCM is also significantly better than CLR ($p < 0.012$) in terms of One Error and testing time. The CLR method shows better predictive performance than the proposed methods in terms of Coverage for the datasets with large number of labels (enron, bibtex and corel5k) and in terms of Average Precision for the enron dataset. For the other datasets, for all evaluation metrics, the proposed methods are statistically better. Comparing to the other methods, TSVM, TSCCM and TSPCCM are 2% to 15% better than the CC method, 2% to 12% better than ML-kNN, 1.5% to 17% better than ML-C4.5 and about 5% better than QWeightedML. The classifier chain method shows better predictive performance than the binary relevance and the label powerset methods.

The results in Table 13 show that the proposed methods (TSVM, TSCCM and TSPCCM) are 2 to 50 times faster (at prediction time) than the calibrated label ranking algorithm with majority voting for all datasets. Also, the proposed methods are 10% to 220% faster than the QWeightedML method for all datasets, except for the corel5k dataset where TSPCCM and TSCCM showed 2 and 2.5 times slower testing time, respectively. The testing time of the BR method is actually the same as the time spent in the testing process for the models located in the first layer of the proposed methods. The testing time of the CC method is slightly longer than the testing time of the BR method as a result of the increase in the feature vector size in each binary relevance model (classifier) of the classifier chain.

The results also show that the ML-C4.5 method is the fastest. The testing time of the ML-kNN method is shorter than the testing time of the problem transformation methods except for the LP method that showed similar testing speed. The intent of including algorithm adaptation methods, as ML-C4.5 and ML-kNN, in the experiments was to show that the proposed methods, that belong to the group of problem transformation methods, can achieve comparable or even better predictive accuracy. However, their performance and computational complexity depend strongly on the type of the base classifier that is used for solving the partial classification problems.

It is obvious that the introduced classifier chain in TSCCM and TSPCCM improves the prediction performance of the TSA. These classifier chain methods outperform the TSVM in all evaluation metrics. In terms of computational complexity in the prediction phase, TSCCM show higher complexity than the TSVM, while TSPCCM show comparable testing times to the TSVM.

TSCCM and TSPCCM show similar predictive performance for the eight multi-label classification problems. Statistically, TSPCCM shows better results than TSCCM in all evaluation metrics, except in Coverage where TSCCM is better but not significantly. In terms of testing time, TSPCCM is significantly better than TSCCM with significance level $p = 0.0117$. So, from the results obtained by the experiments and the analysis made above, we can conclude that the TSPCCM slightly outperforms the TSCCM.

Table 8 also shows the values of the reduction parameter ($r_{real}$) from equation 7 and the values of the parameter $a_{brmf}$. It should be noticed that for smaller values of the reduction parameter $r_{real}$ the testing time of the proposed methods (TSVM, TSCCM and TSPCCM) is close to the testing time of the binary relevance method (BR).

To assess whether the overall differences in performance across the ten different approaches are statistically significant, we employed the corrected Friedman test [22] and the post hoc Nemenyi test [23] as recommended by Demsar [21]. The Friedman test is a non-parametric test for multiple hypotheses testing. It ranks the algorithms according to their performance for each dataset separately, thus the best performing algorithm gets the rank of 1, the second best the rank of 2, etc. In case of ties, it assigns average ranks. Then, the Friedman test compares the average ranks of the algorithms and calculates the Friedman statistic $\chi_F^2$, distributed according to the $\chi_F^2$ distribution with $k - 1$ degrees of freedom ($k$ being the number of algorithms). Iman and Davenport [33] show that the Friedman statistic is undesirably conservative and derive a corrected F-statistic that is distributed according to the F-distribution with $k - 1$ and $(k - 1) \times (N - 1)$ degrees of freedom ($k$ being the number of algorithms and $N$ being the number of datasets).

If a statistically significant difference in the performance is detected, than we can proceed with a post hoc test. The Nemenyi test is used to compare all the classifiers to each other. In this procedure, the performance of two classifiers is significantly different if their average ranks differ more than some critical distance. The critical distance depends on the number of algorithms, the number of datasets and the critical value (for a given significance level - $p$) that is

Table 8: Parameters $t$, $a_{brmf}$ and $r$.

| | $a_{brmf}$ | | | $t$ | | | $r_{real}$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | TSVM | TSCCM | TSPCCM | TSVM | TSCCM | TSPCCM | TSVM | TSCCM | TSPCCM |
| **scene** | 3.337 | 3.337 | 3.337 | 0.02 | 0.02 | 0.02 | 0.260 | 0.260 | 0.260 |
| **yeast** | 7.928 | 7.928 | 7.928 | 0.15 | 0.15 | 0.15 | 0.302 | 0.302 | 0.302 |
| **enron** | 18.46 | 18.46 | 18.37 | 0.03 | 0.03 | 0.035 | 0.117 | 0.117 | 0.116 |
| **emotions** | 3.118 | 3.118 | 3.118 | 0.25 | 0.25 | 0.25 | 0.220 | 0.220 | 0.220 |
| **tmc2007** | 4.93 | 4.93 | 4.93 | 0.1 | 0.1 | 0.1 | 0.042 | 0.042 | 0.042 |
| **medical** | 7.16 | 7.16 | 7.16 | 0.01 | 0.01 | 0.01 | 0.022 | 0.022 | 0.022 |
| **bibtex** | 15.1 | 15.1 | 15.1 | 0.02 | 0.02 | 0.02 | 0.008 | 0.008 | 0.008 |
| **corel5k** | 33.74 | 64.01 | 64.01 | 0.02 | 0.01 | 0.01 | 0.008 | 0.029 | 0.029 |

Table 9: The performance of ten different multi-label classification approaches on eight datasets in terms of the Hamming Loss metric.

| **Algorithm\Dataset** | **scene** | **yeast** | **enron** | **emotions** | **tmc2007** | **medical** | **bibtex** | **corel5k** |
|---|---|---|---|---|---|---|---|---|
| ML-kNN | 0.0989 | 0.1980 | 0.0513 | 0.2937 | 0.0578 | 0.0168 | 0.0139 | **0.0093** |
| ML-C4.5 | 0.1506 | 0.3040 | 0.0636 | 0.3259 | 0.0932 | 0.0150 | 0.0160 | 0.0095 |
| LP | 0.0951 | 0.2056 | 0.0599 | 0.3036 | 0.0285 | **0.0130** | 0.0174 | 0.0166 |
| BR | 0.1177 | 0.2052 | 0.0660 | 0.2714 | **0.0172** | 0.0810 | 0.0122 | 0.0171 |
| CC | 0.1142 | 0.1966 | 0.0645 | 0.2673 | 0.0387 | 0.0796 | 0.0122 | 0.0171 |
| CLR | 0.0963 | 0.1903 | 0.0476 | 0.2566 | 0.0177 | 0.0168 | **0.0121** | 0.0117 |
| QWeightedML | 0.0956 | 0.1909 | 0.0481 | 0.2623 | 0.0263 | 0.0169 | **0.0121** | 0.0117 |
| TSVM | 0.0946 | 0.1906 | 0.0501 | 0.2590 | 0.0228 | 0.0613 | **0.0121** | 0.0119 |
| TSPCCM | 0.0943 | 0.1903 | **0.0401** | **0.2557** | 0.0177 | 0.0610 | **0.0121** | 0.0116 |
| TSCCM | **0.0940** | **0.1900** | 0.0423 | 0.2560 | 0.0177 | 0.0619 | **0.0121** | 0.0112 |

based on the Studentized range statistic and can be found in statistical textbooks.

We present the result from the Nemenyi post hoc test with average rank diagrams as suggested by Demsar [21]. These are given on Figures 2 to 6. The ranks are depicted on the axis, in such a manner that the best ranking algorithms are at the right-most side of the diagram. The algorithms that do not differ significantly (at the significance level of $p = 0.05$) are connected with a line.

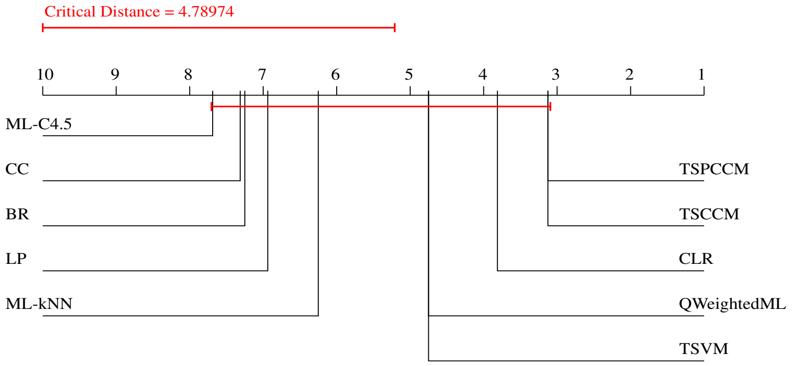

Figure 2: Average ranks diagram comparing the applied algorithms in terms of Hamming Loss.

Table 10: The performance of ten different multi-label classification approaches on eight datasets in terms of the One Error metric.

| **Algorithm\Dataset** | **scene** | **yeast** | **enron** | **emotions** | **tmc2007** | **medical** | **bibtex** | **corel5k** |
|---|---|---|---|---|---|---|---|---|
| ML-kNN | 0.2424 | 0.2344 | 0.2797 | 0.4059 | 0.1903 | 0.2792 | 0.5757 | 0.7060 |
| ML-C4.5 | 0.4038 | 0.2584 | 0.3955 | 0.3762 | 0.1454 | 0.2192 | 0.5292 | 0.7620 |
| LP | 0.2424 | 0.2584 | 0.4801 | 0.4554 | 0.0837 | 0.1982 | 0.7411 | 0.9300 |
| BR | 0.2867 | 0.2388 | 0.2504 | 0.3811 | 0.0480 | 0.2072 | 0.3902 | 0.6640 |
| CC | 0.2951 | 0.2475 | 0.2573 | 0.3861 | 0.0989 | 0.1591 | 0.3862 | 0.6600 |
| CLR | 0.2349 | 0.2334 | 0.2297 | 0.3812 | 0.0411 | 0.1651 | 0.3856 | 0.5920 |
| QWieghtedML | 0.2349 | 0.2301 | 0.2262 | 0.3762 | 0.0821 | 0.1652 | 0.3801 | 0.5920 |
| TSVM | 0.2366 | **0.2300** | **0.2193** | 0.3663 | 0.0631 | 0.1441 | 0.3741 | 0.5980 |
| TSPCCM | 0.2232 | 0.2311 | **0.2193** | **0.3366** | 0.0431 | **0.1261** | 0.3693 | **0.5780** |
| TSCCM | **0.2228** | 0.2308 | 0.2237 | 0.3412 | **0.0402** | 0.1321 | **0.3626** | **0.5780** |

Table 11: The performance of ten different multi-label classification approaches on eight datasets in terms of the Coverage metric.

| **Algorithm\Dataset** | **scene** | **yeast** | **enron** | **emotions** | **tmc2007** | **medical** | **bibtex** | **corel5k** |
|---|---|---|---|---|---|---|---|---|
| ML-kNN | 0.5685 | 6.1413 | 13.181 | 2.4900 | 2.1551 | 2.8348 | 56.266 | 113.05 |
| ML-C4.5 | 1.1086 | 6.8615 | 20.272 | 2.4702 | 2.6713 | 3.8828 | 58.016 | 279.90 |
| LP | 0.8210 | 8.0839 | 30.174 | 2.6831 | 4.3105 | 5.9369 | 82.593 | 332.31 |
| BR | 0.6973 | 6.5485 | 13.245 | 2.4801 | 3.5888 | 2.4744 | 25.173 | 104.00 |
| CC | 0.7081 | 6.4133 | 12.661 | 2.4108 | 1.7531 | 2.0030 | 23.173 | 104.58 |
| CLR | 0.4883 | 6.2758 | **11.519** | 2.4059 | 1.4213 | 2.0570 | **18.551** | **91.624** |
| QWeightedML | 0.7073 | 8.6215 | 20.333 | 2.8465 | 2.0333 | 1.8318 | 57.343 | 206.88 |
| TSVM | 0.4974 | 6.7633 | 14.431 | **2.3960** | 1.7452 | 1.5945 | 36.395 | 193.94 |
| TSPCCM | 0.4740 | 6.0629 | 12.328 | 2.4048 | 1.4324 | **1.4804** | 36.351 | 143.07 |
| TSCCM | **0.4726** | **6.0587** | 12.225 | 2.4060 | **1.4136** | 1.5345 | 36.318 | 110.69 |

Table 12: The performance of ten different multi-label classification approaches on eight datasets in terms of the Average Precision metric.

| **Algorithm\Dataset** | **scene** | **yeast** | **enron** | **emotions** | **tmc2007** | **medical** | **bibtex** | **corel5k** |
|---|---|---|---|---|---|---|---|---|
| ML-kNN | 0.8511 | 0.7584 | 0.6345 | 0.6938 | 0.8442 | 0.7841 | 0.3489 | 0.2655 |
| ML-C4.5 | 0.7335 | 0.7156 | 0.5658 | 0.7166 | 0.8421 | 0.8053 | 0.3922 | 0.1958 |
| LP | 0.8244 | 0.7004 | 0.4485 | 0.6742 | 0.8630 | 0.7767 | 0.2645 | 0.0807 |
| BR | 0.8188 | 0.7548 | 0.6688 | 0.7071 | 0.9073 | 0.8266 | 0.5753 | 0.3032 |
| CC | 0.8159 | 0.7579 | 0.6817 | 0.7137 | 0.9167 | 0.8652 | 0.5762 | 0.2988 |
| CLR | 0.8600 | 0.7685 | **0.7018** | 0.7215 | 0.9630 | 0.8629 | 0.5782 | 0.3520 |
| QWeightedML | 0.8400 | 0.7003 | 0.6543 | 0.6795 | 0.9233 | 0.8617 | 0.4975 | 0.3105 |
| TSVM | 0.8598 | 0.7641 | 0.6970 | 0.7242 | 0.9342 | 0.8816 | 0.5780 | 0.3410 |
| TSPCCM | 0.8667 | 0.7715 | 0.6970 | **0.7323** | 0.9627 | **0.8947** | 0.5801 | 0.3546 |
| TSCCM | **0.8670** | **0.7717** | 0.6933 | 0.7268 | **0.9632** | 0.8880 | **0.5853** | **0.3550** |

Table 13: The testing time of ten different multi-label classification approaches on eight datasets measured in seconds.

| **Algorithm\Dataset** | **scene** | **yeast** | **enron** | **emotions** | **tmc2007** | **medical** | **bibtex** | **corel5k** |
|---|---|---|---|---|---|---|---|---|
| ML-kNN | 13.92 | 5.16 | 3.39 | 0.25 | 230 | 0.29 | 77 | 46 |
| ML-C4.5 | 0.016 | 0.016 | 0.36 | 0.015 | 1.6 | 0.047 | 3.84 | 1.0 |
| LP | 7.00 | 4.34 | 7.62 | 0.21 | 1356 | 0.4 | 508 | 235 |
| BR | 24.12 | 25.01 | 65.17 | 1.02 | 950 | 4.2 | 1370 | 35 |
| CC | 25.05 | 25.12 | 65.67 | 1.06 | 988 | 5.3 | 1410 | 37 |
| CLR | 66.15 | 104.34 | 605 | 2.56 | 6106 | 84 | 83358 | 2020 |
| QWeightedML | 40.32 | 60.39 | 174 | 1.67 | 2534 | 25.2 | 4710 | 119 |
| TSVM | 34.27 | 54.65 | 146 | 1.34 | 1135 | 7.0 | 1564 | 67 |
| TSPCCM | 35.25 | 54.72 | 147 | 1.35 | 1143 | 7.5 | 1593 | 233 |
| TSCCM | 36.68 | 58.42 | 172 | 1.40 | 1737 | 8.8 | 1742 | 328 |

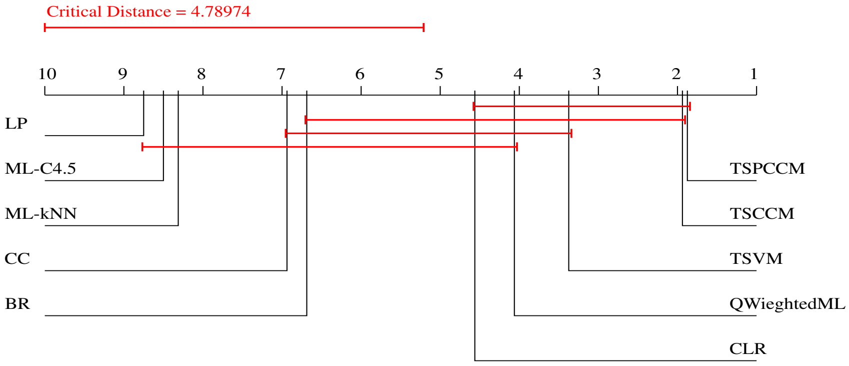

Figure 3: Average ranks diagram comparing the applied algorithms in terms of One Error.

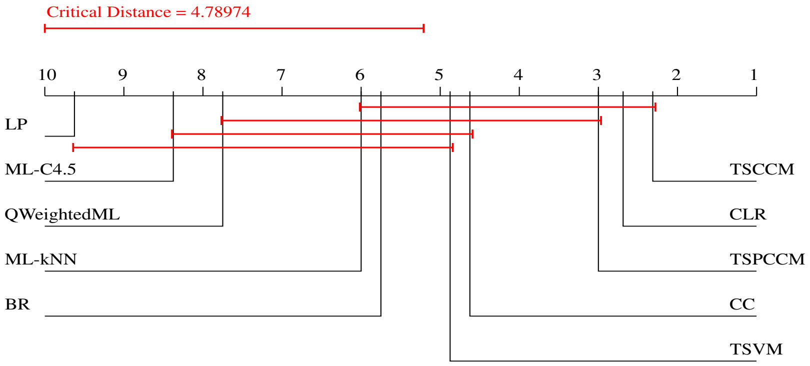

Figure 4: Average ranks diagram comparing the applied algorithms in terms of Coverage.

In the average ranks diagrams, two groups of algorithms are clearly separated. The first contains the pair-wise

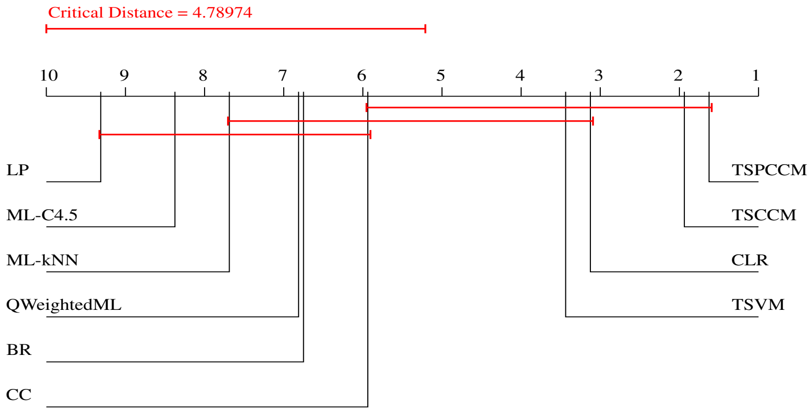

Figure 5: Average ranks diagram comparing the applied algorithms in terms of Average Precision.

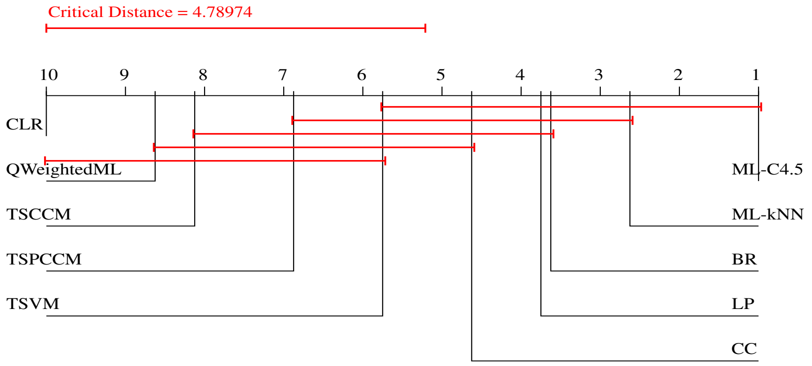

Figure 6: Average ranks diagram comparing the applied algorithms in terms of Testing Time.

methods CLR, QWeightedML and our TSA methods (TSVM, TSCCM and TSPCCM). The second contains the algorithm adaptation methods ML-C4.5 and ML-kNN, LP, BR and CC.

The methods in the first group have better performance in terms of all evaluation metrics. The methods in the second group are faster at testing time. The CC variant of the BR method from the second group has comparable performance to the methods from the first group in terms of Coverage. The QWeightedML method from the first group is only better than the methods in the second group in terms of Hamming Loss and One Error, and worse in terms of the other two metrics.

The separation between the two groups is clearest in terms of the Average Precision and One Error metrics but less clear in terms of Coverage. The statistical tests we use are conservative and the differences in performance for methods within the same group are not significant. However, the best method from the first group is typically significantly better than (most of) the methods from the second group (e.g., TSPCCM and TSCCM in terms of One Error and Average Precision).

The dependence of the predictive performance of the proposed approaches on the different values of the threshold $t$ ($0 \leq t \leq 1$) are shown on Figures 7 to 14 for each dataset separately. The same figures also show the testing times of the methods (TSVM, TSCCM and TSPCCM) as a function of the selected threshold $t$. It can be seen, as $t$ increases, computational efficiency also increases, but performance decreases. One can thus make a trade-off between predictive performance and testing speed by selecting appropriate threshold.

Overall, Coverage varies mostly with $t$, followed by One Error and Average Precision. Hamming Loss shows the least variance. Testing times decrease significantly as $t$ increases, but the rates of decrease vary across the datasets. For the datasets with a smaller number of labels, the decrease is not very sharp and for small values of the threshold $t$ (0.0 - 0.2) the predictive performance of TSVM, TSCCM and TSPCCM changes only slightly, while the testing time decreases significantly (often for more than 40%). For the datasets with a large number of labels (medical - 45, enron - 53, bibtex - 159 and corel5k - 374), testing times decrease sharply between $t = 0.0$ and $t = 0.1$, so the reduction of the testing time of the TSA over the CLR becomes even more notable. The testing time of TSA is ten times shorter than the testing time of CLR for medical, four times for enron, fifty times for bibtex and eight times shorter for corel5k datasets for the corresponding chosen threshold shown in Table 8.

It should be noted that for $t = 1$, the predictive performance of the BR method is better as compared to the predictive performance of the proposed methods. This difference appears because the BR method orders the labels according to the probability association of each label from each model $M_k$, while the proposed methods order the labels by the number of votes that can only be 0 or 1.

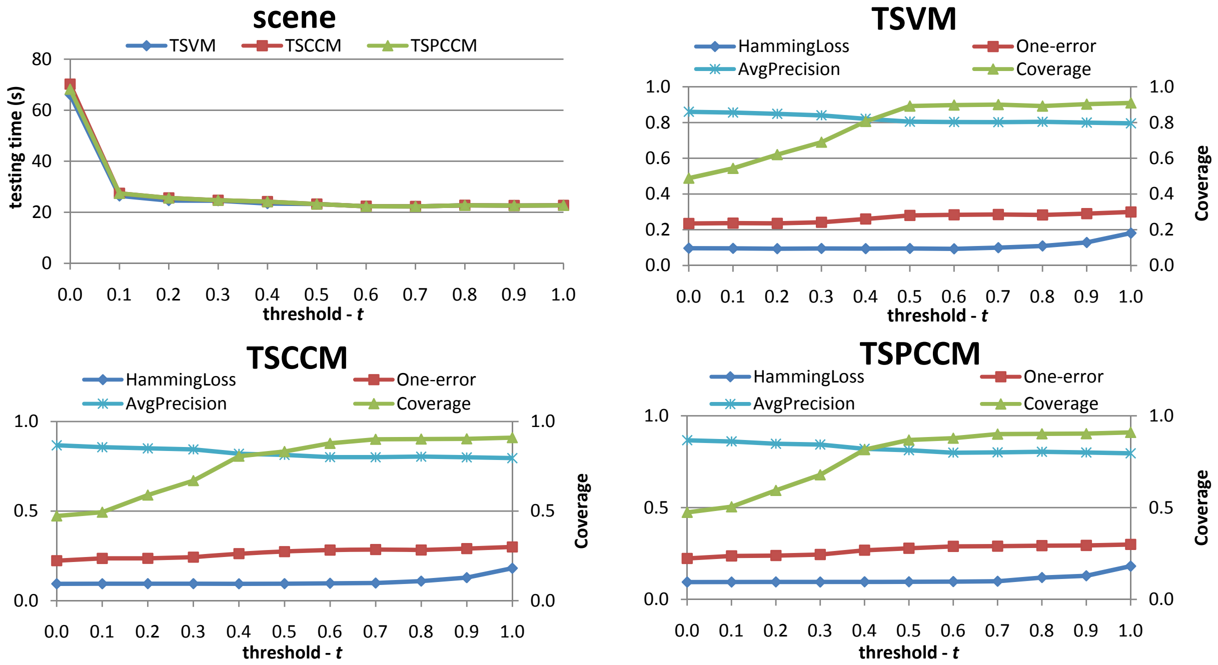

Figure 7: Predictive performance and testing times of TSVM, TSCCM and TSPCCM as functions of the threshold t ($0 \le t \le 1$) for the scene dataset.

## 5. Conclusions

A two stage architecture (TSA) for efficient pair-wise multi-label learning and its three implementations (TSVM, TSCCM and TSPCCM) were presented. The architecture is organized in two layers with binary relevance models in the first layer and pair-wise models in the second layer. Introducing a threshold that dynamically affects the number of pair-wise classifiers consulted in the prediction phase, a significant reduction in the computational complexity was achieved while keeping comparable performance. Utilizing a classifier chain the performance was even improved in the TSCCM and TSPCCM implementations. The proposed architecture offers a possibility of fine grain control of the trade-off between the speed and the predictive performance (of the classifier) by varying the threshold.

The performance of the proposed methods are compared with two algorithm adaptation methods (Multi-Label k-NN and Multi-Label C4.5) and five problem transformation methods (Binary Relevance, Classifier Chain, Calibrated Label Ranking method with majority voting, the Quick Weighted method for pair-wise multi-label learning

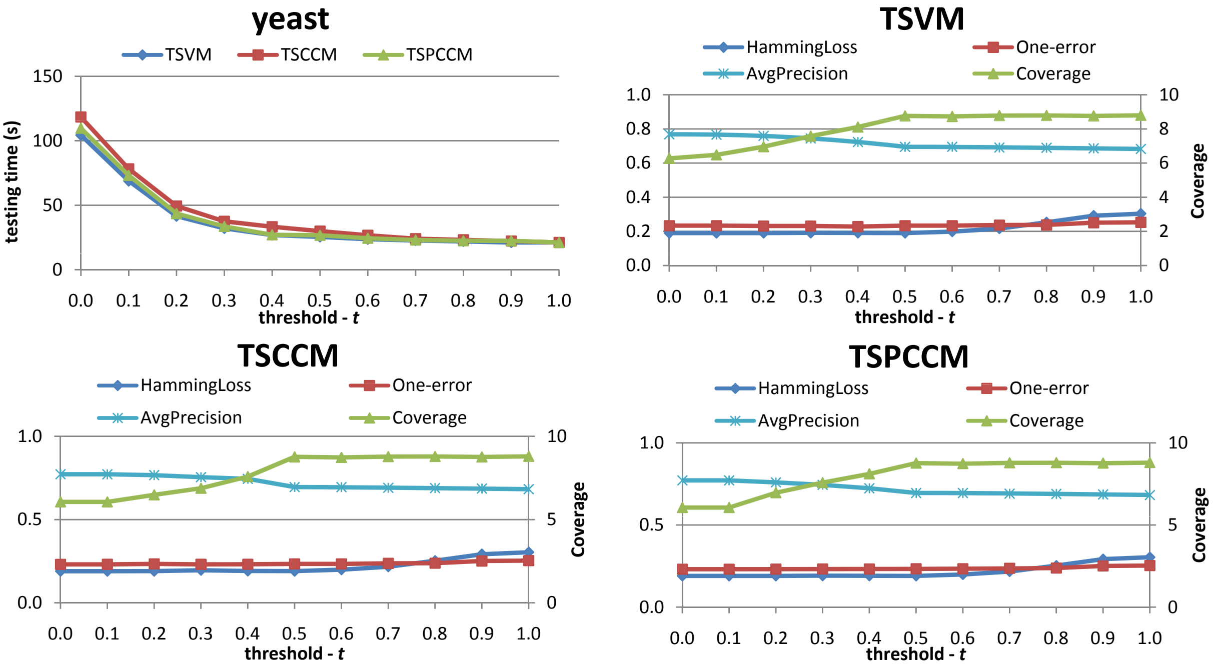

Figure 8: Predictive performance and testing times of TSVM, TSCCM and TSPCCM as functions of the threshold t ($0 \leq t \leq 1$) for the yeast dataset.

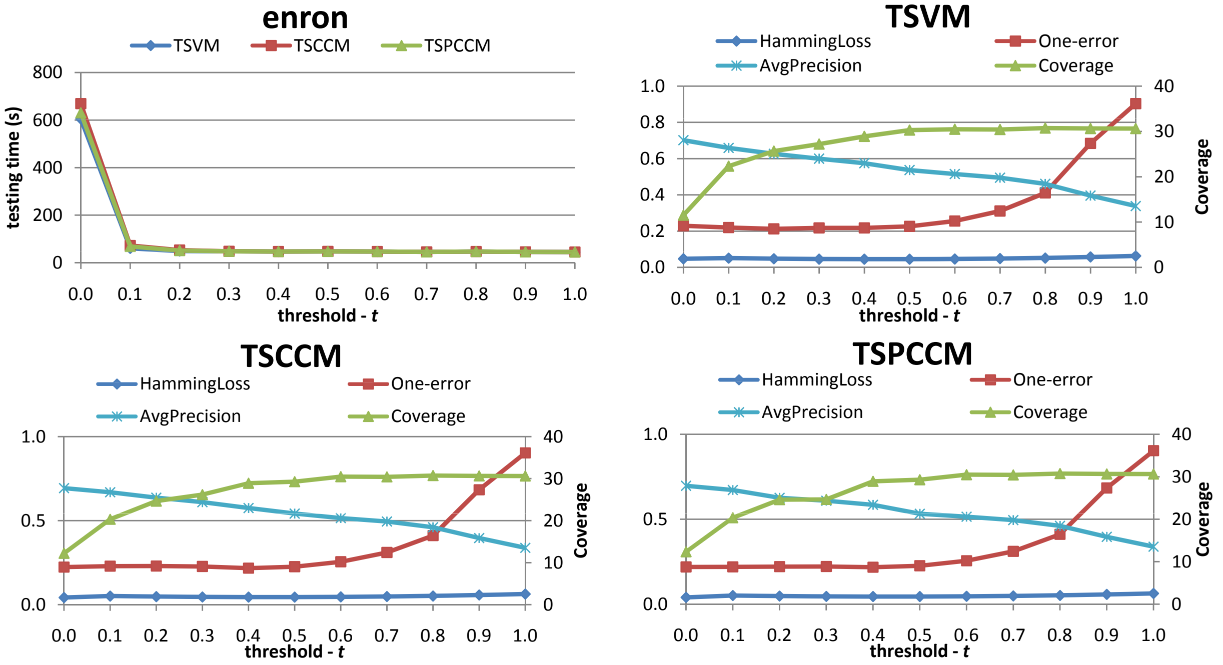

Figure 9: Predictive performance and testing times of TSVM, TSCCM and TSPCCM as functions of the threshold t ($0 \leq t \leq 1$) for the enron dataset.

and the Label Powerset method) on eight different real-world datasets (enron, yeast, scene, emotions, tmc2007, medical, bibtex and corel5k). The TSCCM and the TSPCCM outperform the calibrated label ranking algorithm in terms of

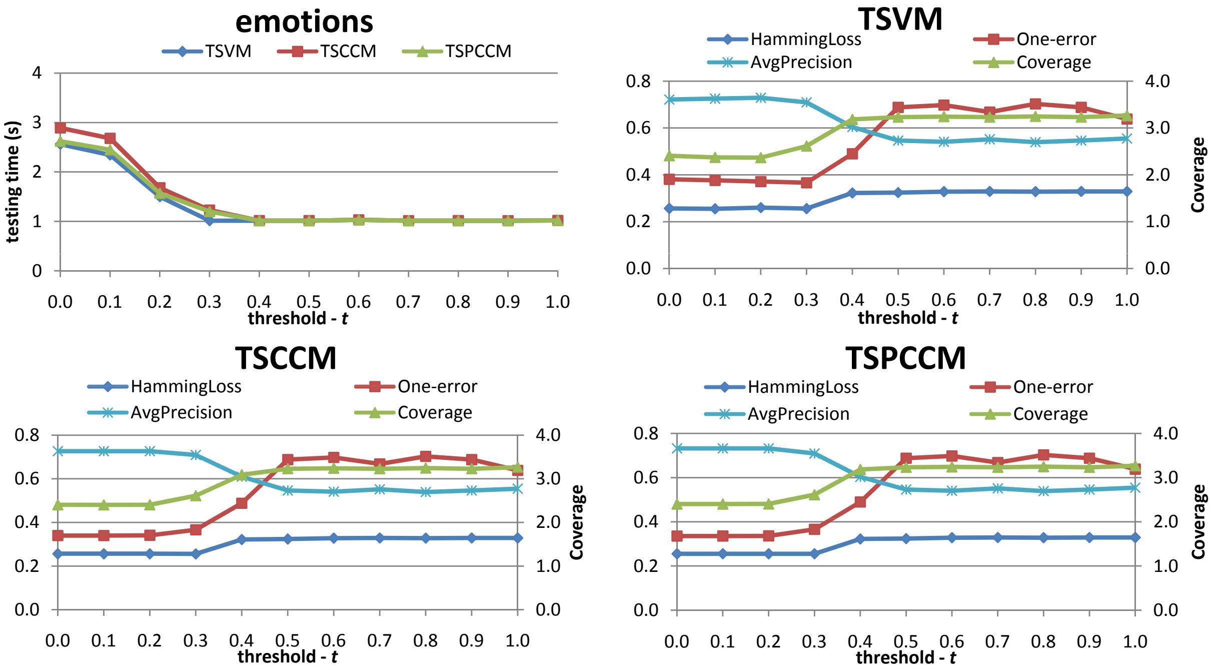

Figure 10: Predictive performance and testing times of TSVM, TSCCM and TSPCCM as functions of the threshold t ($0 \leq t \leq 1$) for the emotions dataset.

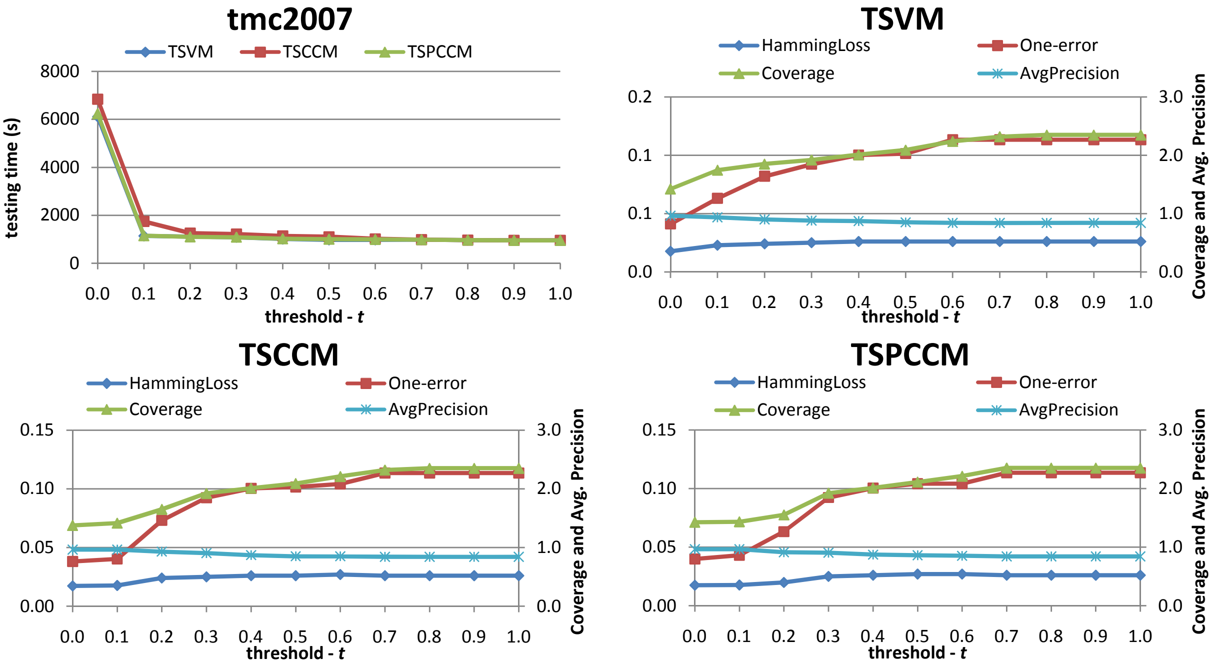

Figure 11: Predictive performance and testing times of TSVM, TSCCM and TSPCCM as functions of the threshold t ($0 \leq t \leq 1$) for the tmc2007 dataset.

predictive performance and also show significantly better predictive performance than the other compared methods. In terms of testing speed the proposed methods (TSVM, TSCCM and TSPCCM) were 2 to 50 times faster than the

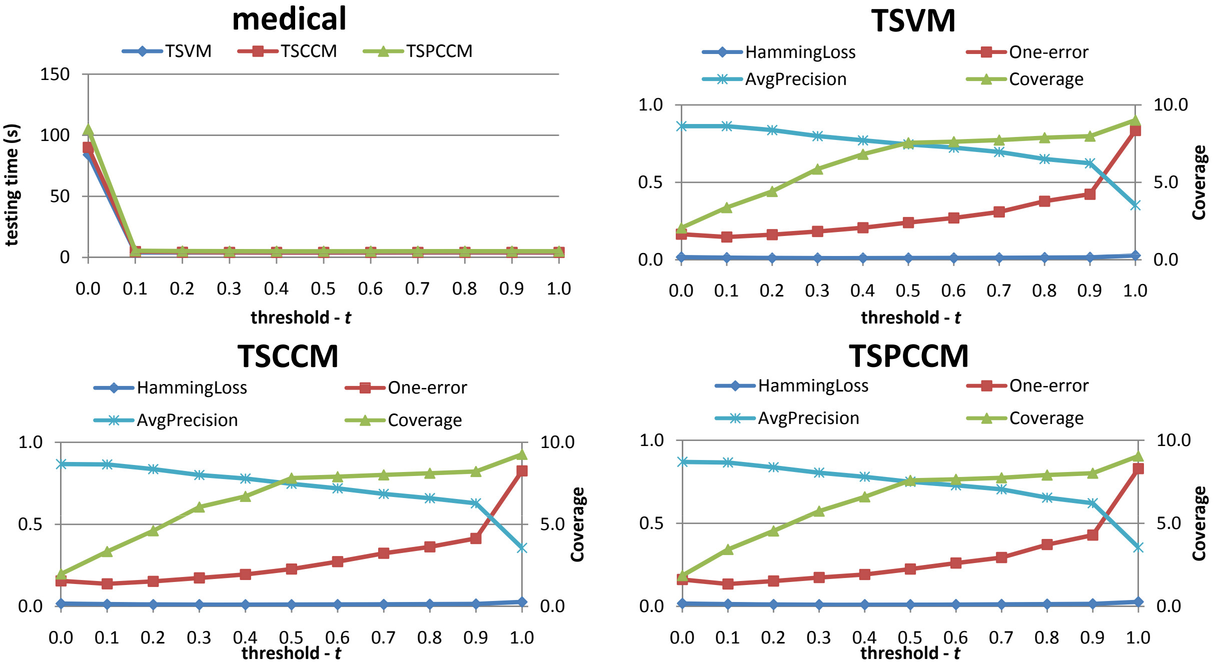

Figure 12: Predictive performance and testing times of TSVM, TSCCM and TSPCCM as functions of the threshold t ($0 \leq t \leq 1$) for the medical dataset.

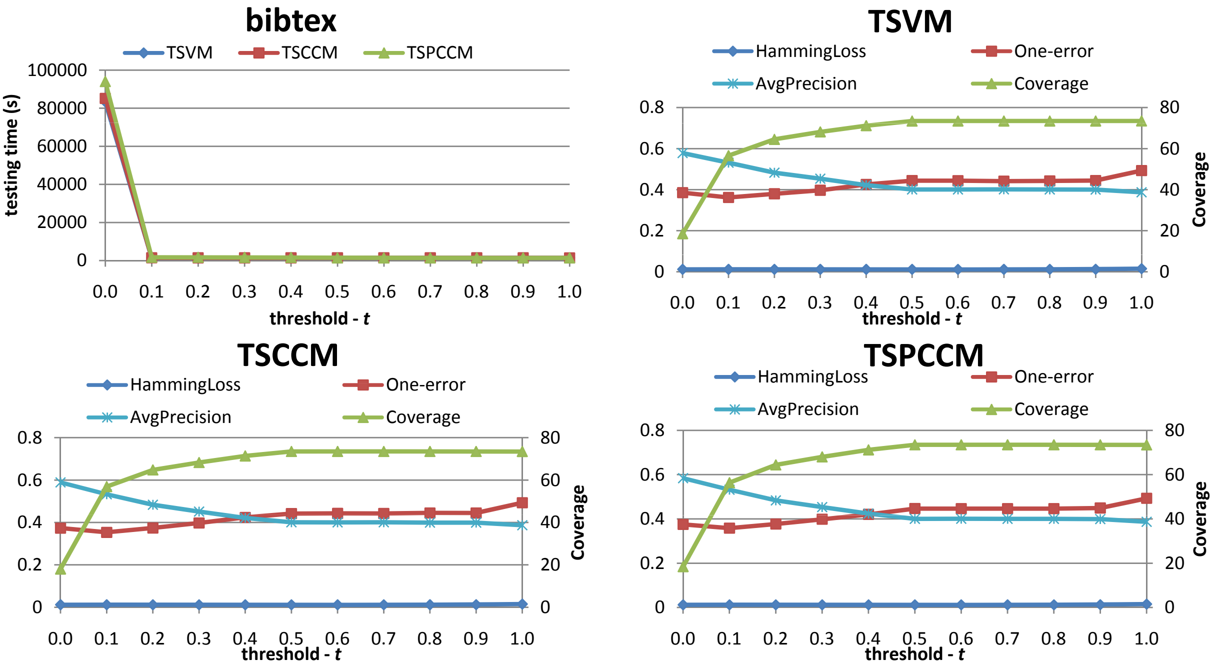

Figure 13: Predictive performance and testing times of TSVM, TSCCM and TSPCCM as functions of the threshold t ($0 \leq t \leq 1$) for the bibtex dataset.

calibrated label ranking algorithm and up to 2.2 times faster than the QWeightedML method. As compared to the binary relevance method (BR), TSVM, TSCCM and TSPCCM show better predictive performance, while the testing

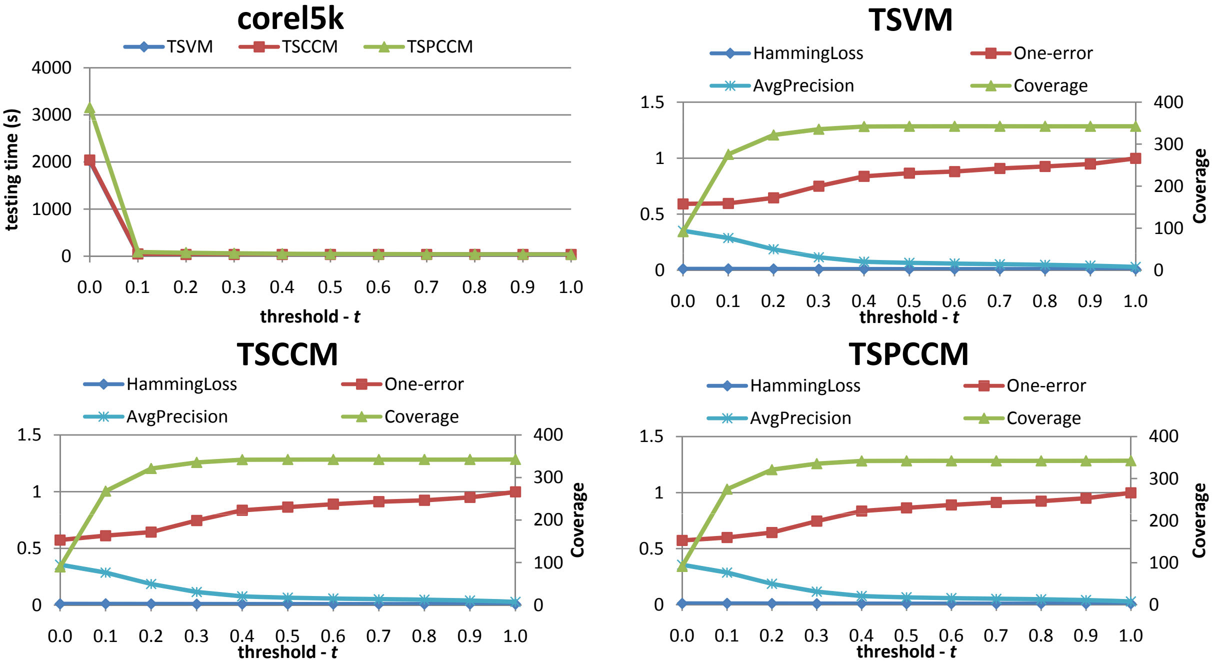

Figure 14: Predictive performance and testing times of TSVM, TSCCM and TSPCCM as functions of the threshold t ($0 \leq t \leq 1$) for the corel5k dataset.

times were always larger than the testing time of the BR method, because of the testing time of the models of the second layer of the architecture. The testing times of TSVM, TSCCM and TSPCCM approach the testing time of the BR method as the value of the reduction parameter $r_{real}$ decreases. In comparison to the algorithm adaptation methods (ML-kNN and ML-C4.5), the proposed methods show better predictive performance, while their computational complexity is higher. Statistically TSPCCM shows better predictive results than TSCCM in all evaluation metrics except in Coverage. Also, its testing time is shorter than the TSCCM.

Let us conclude with some ideas for further work. The original implementations of the two stage architecture involve one global threshold that is determined by cross validation. In our future work, we plan to investigate a new threshold calibration method that establishes a strong connection between the label cardinality of the training data and the predictions for the testing data in the first layer of the architecture. By introducing the label cardinality in this calibration threshold method, we believe that independent thresholds can be introduced for each binary relevance model, leading to improvements in the predictive performance and the computational complexity of the architecture.

## 6. Acknowledgment

The authors would like to thank Dragi Kocev, Department of Knowledge Technologies, Jozef Stefan Institute, Ljubljana for his valuable comments and suggestions.

## References

[1] K. Brinker, J. Fürnkranz, E. Hullermeie, A unified model for multilabel classification and ranking, in: Proc. of the 17th European conference on artificial intelligence, Riva Del Garda, Italy, 2006, pp. 489–493.

[2] G. Tsoumakas, I. Katakis, Multi label classification: An overview, International Journal of Data Warehousing and Mining 3 (3).

[3] M. L. Zhang, Z. H. Zhou, Ml-knn: A lazy learning approach to multi-label learning, Pattern Recognition 40 (7) (2007) 2038–2048.

[4] A. Wieczorkowska, P. Synak, Z. Ras, Multi-label classification of emotions in music, Intelligent Information Processing and Web Mining (2006) 307–315.

[5] E. Spyromitros, G. Tsoumakas, I. Vlahavas, An empirical study of lazy multilabel classification algorithms, Artificial Intelligence: Theories, Models and Applications (2008) 401–406.

[6] K. Crammer, Y. Singer, A family of additive online algorithms for category ranking, Journal of Machine Learning Research 3 (2003) 1025–1058.

[7] M. L. Zhang, Z. H. Zhou, Multi-label neural networks with applications to functional genomics and text categorization, IEEE Transactions on Knowledge and Data Engineering 18 (10) (2006) 1338–1351.

[8] R. E. Schapire, Y. Singer, Boostexter: a boosting-based system for text categorization, Machine Learning 39 (2/3) (2000) 135–168.

[9] F. de Comite, R. Gilleron, 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, Leipzig, Germany, 2003, pp. 35–49.

[10] F. Thabtah, P. Cowling, Y. Peng, Mmac: A new multi-class, multi-label associative classification approach, in: Proc. of the 4th IEEE International Conference on Data Mining, 2004, pp. 217–224.

[11] J. Fürnkranz, Round robin classification, Journal of Machine Learning Research 2 (5) (2002) 721–747.

[12] T. F. Wu, C. J. Lin, R. C. Weng, Probability estimates for multiclass classification by pairwise coupling, Journal of Machine Learning Research 5 (8) (2004) 975–1005.

[13] A. Clare, R. D. King, Knowledge discovery in multi-label phenotype data, in: Proc. of the 5th European Conference on Principles of Data Mining and Knowledge Discovery, Freiburg, Germany, 2001, pp. 42–53.

[14] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, in: Proc. of the 20th European Conference on Machine Learning, 2009, pp. 254–269.

[15] G. Tsoumakas, I. P. Vlahavas, Random k-labelsets: An ensemble method for multilabel classification, in: Proc. of the 18th European Conference on Machine Learning, 2007, pp. 406–417.

[16] J. Read, B. Pfahringer, G. Holmes, Multi-label classification using ensembles of pruned sets, in: Proc. of the 8th IEEE International Conference on Data Mining, 2008, pp. 995–1000.

[17] J. Read, A Pruned Problem Transformation Method for Multi-label classification, in: Proc. of the New Zealand Computer Science Research Student Conference, 2008, pp. 143–150.

[18] S. H. Park, J. Fürnkranz, Efficient pairwise classification, in: Proc. of 18th European Conference on Machine Learning, Warsaw, Poland, 2007, pp. 658–665.

[19] E. L. Mencía, S. H. Park, J. Fürnkranz, Efficient voting prediction for pairwise multi-label classification, Neurocomputing 73 (2010) 1164–1176.

[20] J. Fürnkranz, E. Hullermeier, E. L. Mencía, K. Brinker, Multi-label classification via calibrated label ranking, Machine Learning 73 (2) (2008) 133–153.

[21] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.

[22] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Annals of Mathematical Statistics 11 (1940) 86–92.

[23] P. B. Nemenyi, Distribution-free multiple comparisons, Ph.D. thesis, Princeton University (1963).

[24] M. Boutell, J. Luo, X. Shen, C. Brown, Learning multi-label scene classification, Pattern Recognition 9 (37) (2004) 1757–1771.

[25] P. Duygulu, K. Barnard, N. de Freitas, 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, 2002, pp. 97–112.

[26] A. Elisseeff, J. Weston, A kernel method for multi-labelled classification, Advances in Neural Information Processing Systems 14 (2002) 681–687.

[27] B. Klimt, Y. Yang, The enron corpus: A new dataset for email classification research, in: Proc. of the 15th European Conference on Machine Learning, Pisa, Italy, Springer, 2004, pp. 217–226.

[28] I. Katakis, G. Tsoumakas, I. Vlahavas, Multilabel text classification for automated tag suggestion, in: Proc. of the ECML/PKDD 2008 Discovery Challenge, Antwerp, Belgium, 2008.

[29] A. Srivastava, B. Zane-Ulman, Discovering recurring anomalies in text reports regarding complex space systems, in: Proc. of the IEEE Aerospace Conference, Morgan Kaufmann, 2005, pp. 55–63.

[30] K. Trohidis, G. Tsoumakas, G. Kalliris, I. Vlahavas, Multilabel classification of music into emotions, in: Proc. of International Conference on Music Information Retrieval, Philadelphia, PA, USA, 2008, pp. 320–330.

[31] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, The weka data mining software: An update, software available at `http://www.cs.waikato.ac.nz/ml/weka/`, SIGKDD Explorations 11 (1).

[32] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (2011) 27:1–27:27, software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[33] R. L. Iman, J. M. Davenport, Approximations of the critical region of the friedman statistic, Communications in Statistics (1980) 571–595.