# Obtaining Bipartitions from Score Vectors for Multi-Label Classification

Marios Ioannou, George Sakkas, Grigorios Tsoumakas and Ioannis Vlahavas
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece 54124
Email: {mioannou,gsakkas,greg,vlahavas}@csd.auth.gr

***Abstract*—Multi-label classification is a popular learning task. However, some of the algorithms that learn from multi-label data, can only output a score for each label, so they cannot be readily used in applications that require bipartitions. In addition, several of the recent state-of-the-art multi-label classification algorithms, actually output a score vector primarily and employ one (sometimes simple) thresholding method in order to be able to output bipartitions. Furthermore, some approaches can naturally output both a score vector and a bipartition, but whether a better bipartition can be obtained through thresholding has not been investigated. This paper contributes a theoretical and empirical comparative study of existing thresholding methods, highlighting their importance for obtaining bipartitions of high quality.**

## I. Introduction

Learning from multi-label data has been a very popular research topic during the last years. One of the main reasons is the fact that multi-label data arise in several interesting applications, such as text categorization [1], [2], [3], [4], [5], semantic annotation of images [6], music [7], [8] and video [9], [10], functional genomics [11], [12], [13], [14], [15] and directed marketing [16].

A model that is produced by applying a multi-label learning algorithm to a multi-label dataset can usually give two different kinds of output when presented with a new unknown multi-label instance: a *bipartition* of the labels into relevant and irrelevant with respect to the query instance, and/or a numerical *score* for each label, indicative of their relevance to the query instance. In addition, a *ranking* of the labels according to their relevance with respect to the query instance can be obtained from the numerical scores, after resolving any ties. Some models can output both a score vector and a bipartition. These should be consistent, meaning that there should be no irrelevant labels ranked higher than relevant ones and vice versa. Examples of these types of output and their relationships are presented in Tables I to IV.

TABLE I
A VECTOR OF SCORES, ONE FOR EACH LABEL, INDICATIVE OF THE RELEVANCY OF EACH LABEL TO A TEST INSTANCE $x'$.

| Label 1 | Label 2 | Label 3 | Label 4 | Label 5 | Label 6 |
|---|---|---|---|---|---|
| 0.3 | 0.8 | 0.6 | 0.1 | 0.9 | 0.2 |

TABLE II
A CORRESPONDING RANKING OF THE LABELS ACCORDING TO THEIR RELEVANCY TO $x'$.

| 1st rank | 2nd rank | 3rd rank | 4th rank | 5th rank | 6th rank |
|---|---|---|---|---|---|
| Label 5 | Label 2 | Label 3 | Label 1 | Label 6 | Label 4 |

TABLE III
A BIPARTITION OF THE LABELS INTO RELEVANT AND IRRELEVANT WITH RESPECT TO $x'$ THAT IS *consistent* WITH THE SCORES VECTOR OF TABLE I.

| Relevant labels | Irrelevant labels |
|---|---|
| {Label 2, Label 3, Label 5} | {Label 1, Label 4, Label 6} |

TABLE IV
A BIPARTITION OF THE LABELS INTO RELEVANT AND IRRELEVANT WITH RESPECT TO $x'$ THAT IS *inconsistent* WITH THE SCORES VECTOR OF TABLE I.

| Relevant labels | Irrelevant labels |
|---|---|
| {Label 1, Label 2, Label 5} | {Label 3, Label 4, Label 6} |

For certain applications just a ranking suffices. For example, in computer assisted annotation of multimedia by human annotators, presenting the available concepts ranked by relevance to the object that has to be annotated, would help the annotators a lot, even if no bipartition was available. In fact, in such an application a bipartition might not be desired at all, since in practice it will rarely be perfect, and hence it may hide relevant concepts, especially if it sacrifices recall for precision. However, other applications require obtaining an accurate bipartition, rather than just a ranking. For example, in fully automated annotation, we want a system that is as accurate as possible in its assignment of concepts to an object. As a second example, consider a system that automatically forwards an incoming email to all relevant departments of a company. In this case a ranking isn't useful. Of course, the ideal is having both a ranking and a bipartition, as in this case more information is available for decision making.

We argue that obtaining a bipartition from a score vector, also known as thresholding [17], [18] is an important issue to study for at least three reasons. First, some algorithms just output a ranking [3], [19], so they cannot be used without modifications in applications that require bipartitions. Second, and more important, several of the state-of-the-art multi-label classification algorithms (i.e those that output a bipartition) of the recent literature [20], [21], [22], [23], actually output a score vector primarily and employ one (sometimes simple) thresholding method in order to be able to output bipartitions. Whether the specific thresholding approach used plays a significant role in the quality of the obtained bipartitions has not been thoroughly investigated, neither which thresholding approach is the most appropriate for each of these algorithms. Finally, some approaches [24], [25] can output both a score vector and a consistent bipartition. In these cases, it would be interesting to investigate whether better bipartitions can be obtained via the use of a thresholding strategy.

Motivated from these issues, we here contribute a comparative study of existing thresholding techniques. The study has a theoretical part, where we present six different techniques, abstract their key characteristics discuss their strengths and weaknesses and raise open questions (Section 2), and an empirical part, where we compare their performance on a number of multi-label datasets in conjunction with a variety of multi-label learning algorithms (Section 4). Section 3 presents several multi-label learning algorithms with emphasis on the kind of output they produce and the approach followed for thresholding. Finally, we present the conclusions and open questions of this study (Section 5).

To the best of our knowledge, this is the most extensive work discussing the issue of thresholding for multi-label classification. In the past, thresholding has been studied in [17], but with a focus on text categorization and in the context of just one classification algorithm. The study of [18] focused on the theoretical properties of just a single thresholding strategy and its variations, again in the context of just one classification algorithm.

## II. From Scores to Bipartitions

This section starts with a presentation of six different thresholding techniques of the literature and then continues with a categorization scheme and a discussion of their pros and cons. First, let us introduce some notation. We consider a set of labels $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_q\}$ and a multi-label training set $(x_i, Y_i)$, $i = 1, 2, \dots n$, $Y_i \subseteq \Lambda$. The numerical score predicted for each label $\lambda_j$, given an instance $x'$, is denoted as $s_j(x')$.

*RCut* [17] is one of the simplest approaches for obtaining a bipartition from a scores vector. For each new instance, it outputs the $t$ labels with the highest score. The parameter $t$ can be either specified by the user or automatically tuned using a validation set or $k$-fold cross-validation [17]. In the former case, a logical value is the average number of labels in the training set [26], also called *label cardinality* [27].

Another very simple approach, is to use a single numerical threshold $t$ and given a new instance $x'$ consider each label $\lambda_j$ as relevant, when $s_j(x') \geq t$. We call this approach *OneThreshold* (OT). Several values were explored for the threshold $t$ in [28], while a default value of 0.5 was used in [21]. Cross-validation was used to explore a suitable value for $t$ in [22], while a computationally simpler approach was followed in [23].

*SCut* [17] computes a different threshold $t_j \in [0, 1]$ for each label $\lambda_j$. For each new instance, it outputs those labels with score higher than their corresponding threshold. The thresholds are sequentially tuned independently of one another using a validation set or $k$-fold cross validation. If the measure used for performance evaluation can be decomposed to independent contributions from each label (e.g. Hamming loss, macro-averaged information retrieval measures [27]) then a single pass from each labels is sufficient, otherwise (e.g. micro-averaged or example-based information retrieval measures [27]) the process must reiterate the labels until convergence. Variations of this approach were proposed in [17] and additionally studied theoretically and empirically in [18].

For each label $\lambda_j$, *PCut* [17] computes a different threshold $t_j \in \{0, 1, \dots, n'\}$, where $n'$ is the size of the test set. A label $\lambda_j$ is considered relevant in the test instances with the top $t_j$ scores for that label. Despite that there is a different threshold for each label as in SCut, here these are tuned based on the prior probabilities of relevance for each label and just a single parameter. Similarly to RCut and SCut this parameter can be tuned using a validation set or $k$-fold cross validation.

The main idea of the *MetaLabeler* (Meta) thresholding method [26] is to learn a model for dynamically predicting the size $t$ of the set of relevant labels, given a new unlabeled instance. The $t$ labels with the highest score are then considered as relevant. One important design choice in this approach is the representation of the input space for this learning problem. Three different approaches were proposed and empirically tested in [26]: a) original input-space, b) score vectors, and c) sorted score vectors. Results were in favor of the original input-space. Another important design choice is how to model the learning problem, with two main choices, being regression and multi-class classifications. The authors opted for the multi-class classification approach.

The idea of dynamically thresholding a scores vector, given a new unlabeled instance is actually not that new. A similar approach to MetaLabeler, which was not included in the study of [26], was presented in [12]. The learning problem was formulated differently there. For each example $(x, Y)$, the target value was the numerical threshold $t(x)$ that minimized the quantity:

$$|\lambda_j \in Y : s_j(x) \leq t(x)| + |\lambda_j \in \Lambda \setminus Y : s_j(x) \geq t(x)| \quad (1)$$

The score vectors was the selected input space representation. We call this approach *ThresholdPrediction* (TP).

It is noteworthy that in [12], the regression version of MetaLabeler using the original input space was actually mentioned as a naive solution that would not work and was not studied by the authors. A natural question then to pose is: Which of the two approaches is really more suitable for obtaining bipartitions from score vectors? An even more interesting question is: Which design options (mainly input space representation, but also learning task modeling) work best?

Table V presents the landscape of existing solutions for obtaining bipartitions from scores, in three distinctive dimensions: a) multi-threshold vs single-threshold, b) instance-based vs fixed-threshold, c) rank-based vs score-based. Multi-threshold approaches estimate a separate threshold for each label, whereas single-threshold approaches estimate one threshold for all labels. Instance-based approaches estimate a separate threshold for each instance, or in other words, employ learning for predicting the appropriate threshold for each instance. In contrast, fixed-threshold approaches use the same threshold(s) for all instances. Rank-based approaches estimate a threshold based on a ranking of the labels or the instances, while score-based approaches estimate a threshold based on the original scores.

TABLE V
THE LANDSCAPE OF EXISTING SOLUTIONS FOR OBTAINING A BIPARTITION FROM SCORES IN THREE DISTINCTIVE DIMENSIONS

| Solution | Multi-threshold vs Single-threshold | Instance-based vs Fixed-threshold | Rank-based vs Score-based |
|---|---|---|---|
| OneThreshold | Single-threshold | Fixed-threshold | Score-based |
| RCut | Single-threshold | Fixed-threshold | Rank-based |
| PCut | Multi-threshold | Fixed-threshold | Rank-based |
| SCut | Multi-threshold | Fixed-threshold | Score-based |
| ThresholdPrediction | Single-threshold | Instance-based | Score-based |
| MetaLabeler | Single-threshold | Instance-based | Rank-based |

A disadvantage of the four fixed-threshold approaches that optimize the thresholds based on a specific evaluation measure is that they are bound to this measure. In contrast, instance-based approaches work independently of a specific evaluation measure. However, they are dependent on the classification or regression algorithm used underneath, which needs to be selected and properly tuned. This is more complex compared to simply selecting an evaluation measure to optimize.

*PCut* requires the existence of a complete test set in order to provide the bipartitions of each test instance. Therefore, it is impractical for online multi-label classification applications, such as automated tag suggestion [29], [30], query categorization [26] and email categorization. Note that the same problem holds for the computationally simpler approach to tuning *OneThreshold*, presented in [23].

## III. Multi-Label Learning Algorithms and their Output

Two methods that only output rankings are the Multi-class Multi-label Perceptron (MMP) [3] and Ranking by Pairwise Comparison (RPC) [19]. MMP is a family of online algorithms for label ranking from multi-label data based on the perceptron algorithm. It maintains one perceptron for each label, but weight updates for each perceptron are performed so as to achieve a perfect ranking of all labels. RPC learns pairwise binary models for all pairs of labels. Given a new instance, all binary classifiers are invoked, and a score vector is obtained by counting the votes received by each label. Both MMP and RPC cannot output a bipartition without the aid of a thresholding approach.

Random $k$-Labelsets (RA$k$EL) [28], [21], Ensembles of Pruned Sets (EPS) [22] and Ensembles of Classifier Chains (ECC) [23] are three ensemble algorithms for multi-label classification. All three primarily produce a score vector by averaging the zero-one predictions of their multiple multi-label models. To obtain bipartitions, all three employ variations of the simple *OneThreshold* approach, as discussed in the previous Section.

RA$k$EL and EPS are based on the Label Powerset (LP) approach [6], which learns a multi-class classifier with the classes being all distinct label combinations that appear in the training set. LP primarily outputs bipartitions. RA$k$EL builds several LP models on random subsets of the original set of labels.

EPS uses subsampling (63%) to construct several models using the Pruned Sets approach [31], [23]. Pruned Sets is an extension of LP that improves upon it by pruning highly infrequent label combinations and at the same time reintroducing their information in the training set as multiple frequent label combinations.

ECC builds several *classifier chains*, each chain consisting of a sequence of $q$ binary models that predict a label, given the original feature vector and the true values of the labels that precede it in the sequence. Predictions for a classifier chain are formed sequentially.

BP-MLL [20] is a neural network approach for multi-label classification based on the popular back-propagation algorithm. The main modification is the introduction of a new global error function that takes multiple labels into account. As the output units used hyperbolic tangent functions, a natural threshold value for the score of each label was zero. However, BP-MLL incorporated the solution in [12] (*ThresholdPrediction*) for obtaining a bipartition.

ML-$k$NN [24] is a lazy learning algorithm that uses the maximum a posteriori principle in order to determine the label set of the test instance, based on prior and posterior probabilities for the frequency of each label within the $k$ nearest neighbors. It outputs both a vector of estimated posterior probabilities for the positive class of each label and a consistent bipartition. Since it is based on a probabilistic framework, we can think of an implicit threshold of 0.5 being applied on the score of each label. Although there is no actual need here to obtain a bipartition, one question that can be posed is whether a better bipartition can be obtained from the score vector, using a thresholding strategy.

The Binary Relevance (BR) approach learns a separate binary classification model for each label and thus naturally outputs a bipartition. If the algorithm used to learn the binary classification model can output a score for each class (e.g. probability), then a score vector can be obtained as a by-product by considering the scores for the positive class of each label. Similarly to ML-$k$NN, it would be interesting to investigate whether a better bipartition can be obtained from this score vector.

Calibrated Label Ranking (CLR) [25] is in practice a combination of BR and RPC (a description of RPC has been given at the start of this Section), which allows for naturally outputting a consistent bipartition and scores vector, by introducing a virtual label whose votes act as a threshold. Therefore, CLR does not require a separate method to obtain a bipartition. However, it would be interesting to see whether the bipartition obtained from the scores (of RPC or CLR) and a thresholding strategy is better compared to the one obtained through the virtual label trick.

## IV. Setup of Experiments

### A. Datasets

Table VI presents the 5 multi-label datasets[1] that were used in the experiments. It includes statistics, such as the number of examples, the number of features, the number of labels, the average number of relevant labels (label cardinality) and the number of distinct label combinations (labelsets).

TABLE VI
Multi-label datasets used in the experiments.

| name | examples | features | labels | cardinality | labelsets |
|---|---|---|---|---|---|
| emotions | 593 | 72 | 6 | 1.869 | 27 |
| enron | 1702 | 1001 | 53 | 3.378 | 753 |
| medical | 978 | 1449 | 45 | 1.245 | 94 |
| scene | 2407 | 294 | 6 | 1.074 | 15 |
| yeast | 2417 | 103 | 14 | 4.237 | 198 |

The *emotions* dataset [8] contains 593 songs, annotated with one or more out of 6 labels corresponding to evoked emotions, such as *amazed-suprised* and *sad-lonely*. Each song is described with 72 features extracted from the audio signal.

The *enron* dataset is based on a collection of email messages exchanged between the Enron Corporation employees, which was made available during a legal investigation. It contains 1702 email messages that were categorized into 53 topic categories, such as *company strategy*, *humor* and *legal advice*, by the UC Berkeley Enron Email Analysis Project[2].

The *medical* dataset[3] is based on the data made available during the Computational Medicine Center's 2007 Medical Natural Language Processing Challenge[4]. It consists of 978 clinical free text reports labeled with one or more out of 45 disease codes.

The *scene* dataset contains 2407 images annotated with up to 6 concepts such as *beach*, *mountain* and *field* [6]. Each image is described with 294 visual features.

The *yeast* dataset [12] contains micro-array expressions and phylogenetic profiles for 2417 yeast genes. Each gene is annotated with a subset of 14 functional categories (e.g. *metabolism*, *energy*, etc) from the top level of the functional catalogue (FunCat).

### B. Multi-Label Learning Algorithms

We investigate the effect of thresholding on the following multi-label learning algorithms: ML-$k$NN, BP-MLL, BR and CLR.

The parameters of ML-$k$NN were set as specified in [24]: the number of neighbors was set to 10 and the smoothing factor to 1. In addition, the feature space of each dataset was normalized prior to the application of the algorithm.

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

[2]http://bailando.sims.berkeley.edu/enron_email.html

[3]Originally obtained from http://www.cs.waikato.ac.nz/∼jmr30/

[4]http://www.computationalmedicine.org/challenge/index.php

The parameters of BP-MLL were configured as specified in [20]: the learning rate was set to 0.05, the number of epochs to 100 and the number of hidden units to 20% of the input units.

As BR, and CLR are problem transformation approaches [27], we must specify the binary classification algorithm used underneath. For BR we used an ensemble of decision trees produced via bagging [32], as bagging is known to provide a relatively smooth probabilistic output, which in turn offers more flexibility to the threshold selection strategies. A single decision tree was used underneath CLR.

### C. Thresholding Strategies

We experiment with all thresholding strategies presented in Section 2, with the exception of PCut. Similarly to [26], PCut was excluded from the study because of its important inability to provide bipartitions in an online fashion.

In fixed-threshold approaches we experiment with two different approaches to performance evaluation: a) using the training set (train), and b) using 5-fold cross-validation (5-cv). For RCut, we also experiment with the simpler approach of setting the threshold to the cardinality of the training set.

In instance-based approaches, we experiment with all three different representation used in [26]: a) original input-space (content), b) score vectors (scores), and c) sorted score vectors (ranks). We employ decision tree learning algorithms underneath: the C4.5 algorithm [33] for classification and the M5P algorithm [33], [34] for regression.

### D. Evaluation

10-fold cross-validation is used to evaluate the performance of the various algorithms. Several performance measures exist for the different kinds of output of multi-label learners [27]. Here, we are interested in evaluating the quality of bipartitions. One popular measure for this purpose is the average Hamming loss [2] across all test examples. The Hamming loss between a predicted set of labels $Z_i$ and an actual set of labels $Y_i$ is defined as:

$$\text{Hamming-Loss} = \frac{|Y_i \triangle Z_i|}{q}$$

where $\triangle$ stands for the symmetric difference of two sets, which is the set-theoretic equivalent of the exclusive disjunction (XOR operation) in Boolean logic.

Another applicable measure is *subset accuracy*, which is equal to the zero-one loss for the single-label classification task of predicting the exact label subset. This, however, is a very strict measure. In addition, it cannot discriminate between label subsets that are close to the correct one and label subsets that are far from it.

Several other example-based and label-based measures exist for evaluating bipartitions [27]. We would like to mention a technical problem of some of these measures, in particular those related to precision (and as a result f-measure), which appears stronger in this particular type of study: Their value can become undefined for some test examples or labels. For instance, example-based precision requires division with $|Z_i|$, which under some high thresholding values becomes zero. A similar problem occurs for label-based precision.

For clarity of presentation and the aforementioned considerations, we decided to present results only in terms of average Hamming loss.

The experiments were performed using the Weka library [35] in conjunction with the Mulan library [28]. They can be replicated by executing file `ICTAI2010.java` located in the `experiments` package.

## V. Results and Discussion

Tables VII, VIII, IX and X presents the average Hamming loss of ML-$k$NN, BP-MLL, BR and CLR respectively, under the different thresholding strategies for all datasets. The last column of these tables shows the average rank of all strategies and variations across all datasets.

### A. *ML-kNN*

We note the following interesting points with respect to thresholding ML-$k$NN:

- Using 5-fold cross-validation, OT achieves slightly better results compared to using the training set.
- Using the training set leads to equal results compared to using 5-fold cross-validation for RCut. Both these variations are slightly better compared to using the cardinality of the training set.
- For SCut, using 5-fold cross-validation leads to slightly better results compared to using the training set.
- For Meta and TP, using the training set leads generally to better results compared to using 5-fold cross-validation.
- For Meta using regression leads to slightly better results compared to using classification.
- For TP, scores leads to slightly better results than ranks and content.
- Based on the average ranks, OT leads to the best results, followed by variations of TP (content and scores using the training set).
- None of the strategies manage to outperform the default strategy of ML-$k$NN. This algorithm does not require a special thresholding strategy to obtain bipartitions.

### B. *BP-MLL*

We note the following interesting points with respect to thresholding BP-MLL:

- Using 5-fold cross-validation, OT achieves slightly better results compared to using the training set.
- For RCut, using the training set leads to slightly better results compared to using 5-fold cross-validation, which in turn leads to better results compared to using the cardinality of the training set.
- Using the training set, leads to better results for SCut, compared to using 5-fold cross-validation.
- In the case of Meta, the two best results are obtained using the scores representation and the training set.
- The two worst results for TP are obtained using ranks.
- BP-MLL depends very strongly on a thresholding strategy to obtain bipartitions. This can be explained from the fact that BP-MLL is designed so as to minimize the ranking performance and not the quality of bipartitions. Its global error function is based on the ranking loss.
- Designing a high quality ranker and combining it with a high quality thresholding strategy can lead to high quality bipartitions.
- Based on the average ranks, OT leads to the best results, followed by RCut (train and 5-cv).

### C. *BR*

We note the following interesting points with respect to thresholding BR:

- Using 5-fold cross-validation, OT achieves slightly better results compared to using the training set.
- Using 5-fold cross-validation, RCut achieves equal or better results compared to the other two variations.
- Using 5-fold cross-validation, SCut achieves equal or better results compared to using the training set.
- In Meta, the two best results use ranks, while the three worst results use classification with 5-fold cross-validation.
- For TP, the two best results use the training set, while using scores leads to better results compared to ranks and content.
- Based on the average ranks, OT (5-cv) leads to the best results, followed by TP (scores, train).

### D. *CLR*

We note the following interesting points with respect to thresholding CLR:

- For OT, using the training set is slightly better compared to using 5-fold cross-validation.
- All variations of RCut perform very similarly across all datasets.
- Using 5-fold cross-validation, SCut achieves better results compared to using the training set in all datasets.
- For Meta, using scores leads to better results compared to using content or ranks in most cases. In addition, using the training set leads to better results compared to using 5-fold cross-validation in most cases.
- For TP, the best two results are using the training set.
- None of the thresholding strategies can clearly beat the custom strategy of CLR. This means that the bipartitions output by CLR are of high quality by default. Still, however, using the training set OT manages to beat CLR in three datasets and lose only in one.
- Among the competing methods, good overall results are achieved by OT and one variation of TP (content, train).

### E. *General Comments*

A first general comment is that OT performs very well compared to the rest of the strategies across all multi-label learning algorithms and datasets. Taking also its simplicity into account, it can be considered as a very practical approach. One

TABLE VII
HAMMING LOSS OF ML-$k$NN UNDER DIFFERENT THRESHOLDING STRATEGIES IN ALL DATASETS.

| strategy | parameters | emotions | scene | yeast | enron | medical | rank |
|---|---|---|---|---|---|---|---|
| default | - | 0.195±0.024 | 0.086±0.008 | 0.193±0.012 | 0.052±0.002 | 0.015±0.002 | 2.6 |
| OT | train | 0.192±0.027 | 0.088±0.010 | 0.194±0.011 | 0.053±0.002 | 0.015±0.002 | 4.8 |
| OT | 5-cv | 0.194±0.025 | 0.086±0.010 | 0.193±0.013 | 0.053±0.002 | 0.015±0.002 | 2.6 |
| RCut | cardinality | 0.224±0.027 | 0.087±0.008 | 0.200±0.014 | 0.059±0.002 | 0.016±0.002 | 13.9 |
| RCut | train | 0.224±0.027 | 0.087±0.008 | 0.200±0.014 | 0.056±0.002 | 0.016±0.002 | 12.7 |
| RCut | 5-cv | 0.224±0.027 | 0.087±0.008 | 0.200±0.014 | 0.056±0.001 | 0.016±0.002 | 12.7 |
| SCut | train | 0.194±0.025 | 0.086±0.010 | 0.200±0.014 | 0.072±0.009 | 0.045±0.022 | 13.9 |
| SCut | 5-cv | 0.191±0.023 | 0.086±0.009 | 0.194±0.012 | 0.060±0.006 | 0.063±0.020 | 10.6 |
| Meta | regression, content, train | 0.220±0.024 | 0.089±0.009 | 0.206±0.012 | 0.060±0.002 | 0.015±0.002 | 14.8 |
| Meta | regression, scores, train | 0.208±0.029 | 0.087±0.009 | 0.204±0.012 | 0.059±0.002 | 0.016±0.002 | 12.3 |
| Meta | regression, ranks, train | 0.212±0.027 | 0.087±0.009 | 0.204±0.012 | 0.058±0.002 | 0.016±0.002 | 12.5 |
| Meta | regression, content, 5-cv | 0.228±0.016 | 0.092±0.010 | 0.210±0.013 | 0.060±0.003 | 0.016±0.003 | 20 |
| Meta | regression, scores, 5-cv | 0.227±0.016 | 0.090±0.010 | 0.205±0.012 | 0.061±0.003 | 0.017±0.003 | 20.5 |
| Meta | regression, ranks, 5-cv | 0.227±0.016 | 0.091±0.010 | 0.204±0.012 | 0.061±0.003 | 0.016±0.003 | 19.1 |
| Meta | classification, content, train | 0.214±0.029 | 0.092±0.010 | 0.220±0.009 | 0.059±0.003 | 0.015±0.001 | 16.4 |
| Meta | classification, scores, train | 0.208±0.022 | 0.087±0.009 | 0.213±0.011 | 0.058±0.002 | 0.016±0.002 | 12.9 |
| Meta | classification, ranks, train | 0.212±0.027 | 0.087±0.009 | 0.212±0.012 | 0.058±0.002 | 0.016±0.002 | 13.5 |
| Meta | classification, content, 5-cv | 0.223±0.020 | 0.096±0.010 | 0.224±0.010 | 0.060±0.002 | 0.016±0.003 | 20.4 |
| Meta | classification, scores, 5-cv | 0.227±0.016 | 0.090±0.010 | 0.216±0.012 | 0.060±0.003 | 0.017±0.003 | 20.7 |
| Meta | classification, ranks, 5-cv | 0.217±0.028 | 0.090±0.010 | 0.217±0.013 | 0.061±0.003 | 0.017±0.003 | 20.3 |
| TP | content, train | 0.203±0.026 | 0.087±0.011 | 0.196±0.011 | 0.055±0.002 | 0.017±0.004 | 9.4 |
| TP | scores, train | 0.202±0.024 | 0.090±0.010 | 0.196±0.012 | 0.055±0.002 | 0.016±0.002 | 9.1 |
| TP | ranks, train | 0.207±0.025 | 0.094±0.011 | 0.197±0.012 | 0.055±0.002 | 0.018±0.002 | 13.8 |
| TP | content, 5-cv | 0.214±0.021 | 0.091±0.011 | 0.198±0.012 | 0.056±0.002 | 0.018±0.004 | 15.7 |
| TP | scores, 5-cv | 0.210±0.026 | 0.091±0.009 | 0.197±0.011 | 0.055±0.001 | 0.016±0.003 | 11.2 |
| TP | ranks, 5-cv | 0.211±0.022 | 0.096±0.011 | 0.198±0.012 | 0.055±0.002 | 0.017±0.003 | 14.6 |

TABLE VIII
HAMMING LOSS OF BP-MLL UNDER DIFFERENT THRESHOLDING STRATEGIES IN ALL DATASETS.

| strategy | parameters | emotions | scene | yeast | enron | medical | rank |
|---|---|---|---|---|---|---|---|
| default | - | 0.224±0.022 | 0.334±0.067 | 0.259±0.020 | 0.286±0.066 | 0.707±0.026 | 24.8 |
| OT | train | 0.213±0.027 | 0.177±0.004 | 0.223±0.012 | 0.064±0.002 | 0.028±0.001 | 6.5 |
| OT | 5-cv | 0.204±0.023 | 0.189±0.029 | 0.216±0.012 | 0.064±0.002 | 0.028±0.001 | 6.1 |
| RCut | cardinality | 0.221±0.023 | 0.180±0.022 | 0.212±0.012 | 0.096±0.012 | 0.049±0.002 | 10.4 |
| RCut | train | 0.234±0.028 | 0.175±0.014 | 0.211±0.012 | 0.063±0.002 | 0.028±0.001 | 7.1 |
| RCut | 5-cv | 0.231±0.031 | 0.177±0.015 | 0.211±0.012 | 0.064±0.002 | 0.028±0.001 | 7.8 |
| SCut | train | 0.216±0.021 | 0.151±0.005 | 0.225±0.010 | 0.055±0.002 | 0.073±0.005 | 9.1 |
| SCut | 5-cv | 0.204±0.030 | 0.205±0.028 | 0.209±0.011 | 0.171±0.056 | 0.526±0.049 | 14.5 |
| Meta | regression, content, train | 0.215±0.016 | 0.182±0.023 | 0.218±0.014 | 0.093±0.010 | 0.053±0.002 | 12 |
| Meta | regression, scores, train | 0.212±0.017 | 0.179±0.025 | 0.219±0.013 | 0.093±0.013 | 0.050±0.002 | 9.1 |
| Meta | regression, ranks, train | 0.213±0.024 | 0.190±0.020 | 0.216±0.010 | 0.093±0.010 | 0.050±0.002 | 10.3 |
| Meta | regression, content, 5-cv | 0.222±0.024 | 0.199±0.023 | 0.226±0.012 | 0.097±0.011 | 0.053±0.003 | 16.3 |
| Meta | regression, scores, 5-cv | 0.229±0.025 | 0.177±0.016 | 0.217±0.015 | 0.103±0.006 | 0.049±0.002 | 11.8 |
| Meta | regression, ranks, 5-cv | 0.236±0.027 | 0.173±0.016 | 0.218±0.012 | 0.098±0.008 | 0.049±0.002 | 11.9 |
| Meta | classification, content, train | 0.219±0.018 | 0.184±0.017 | 0.231±0.010 | 0.092±0.010 | 0.052±0.002 | 14.5 |
| Meta | classification, scores, train | 0.199±0.015 | 0.181±0.035 | 0.231±0.011 | 0.092±0.011 | 0.050±0.001 | 10.1 |
| Meta | classification, ranks, train | 0.214±0.028 | 0.180±0.024 | 0.230±0.010 | 0.092±0.005 | 0.051±0.002 | 11.5 |
| Meta | classification, content, 5-cv | 0.220±0.021 | 0.177±0.010 | 0.237±0.013 | 0.098±0.006 | 0.052±0.003 | 15.8 |
| Meta | classification, scores, 5-cv | 0.230±0.021 | 0.182±0.021 | 0.231±0.013 | 0.096±0.012 | 0.050±0.002 | 15.8 |
| Meta | classification, ranks, 5-cv | 0.231±0.023 | 0.175±0.026 | 0.237±0.009 | 0.096±0.009 | 0.049±0.002 | 14.2 |
| TP | content, train | 0.210±0.024 | 0.290±0.023 | 0.231±0.013 | 0.112±0.016 | 0.073±0.028 | 16.9 |
| TP | scores, train | 0.215±0.027 | 0.284±0.045 | 0.229±0.014 | 0.123±0.014 | 0.081±0.011 | 17.9 |
| TP | ranks, train | 0.218±0.029 | 0.310±0.041 | 0.231±0.011 | 0.129±0.039 | 0.084±0.020 | 20.7 |
| TP | content, 5-cv | 0.212±0.016 | 0.277±0.051 | 0.233±0.018 | 0.132±0.039 | 0.082±0.043 | 18.6 |
| TP | scores, 5-cv | 0.212±0.032 | 0.280±0.027 | 0.227±0.008 | 0.158±0.060 | 0.121±0.053 | 17.7 |
| TP | ranks, 5-cv | 0.217±0.023 | 0.297±0.052 | 0.228±0.013 | 0.129±0.047 | 0.121±0.038 | 19.6 |

explanation for this, could be that having just one parameter, it does not overfit as much as the rest of the methods. RCut, also has a single parameter, but suffers from another limitation: that of outputting a fixed number of labels, whereas OT is more flexible in this respect. In our future work, we will explore the behavior of these methods in larger datasets, to see whether the rest of the methods are more competitive in this case.

A second general comment is that thresholding strategies can improve the quality of bipartitions of most of the algorithms in most of the datasets. Therefore, they should be considered as a crucial component of multi-label classification algorithms. Ignoring the issue of thresholding may lead us to wrong conclusions, concerning the ability of multi-label learners to output bipartitions.

TABLE IX
HAMMING LOSS OF BR UNDER DIFFERENT THRESHOLDING STRATEGIES IN ALL DATASETS.

| strategy | parameters | emotions | scene | yeast | enron | medical | rank |
|---|---|---|---|---|---|---|---|
| default | - | 0.205±0.026 | 0.092±0.007 | 0.204±0.009 | 0.048±0.002 | 0.011±0.001 | 7 |
| OT | train | 0.207±0.025 | 0.092±0.008 | 0.207±0.011 | 0.049±0.003 | 0.010±0.001 | 7.1 |
| OT | 5-cv | 0.202±0.029 | 0.091±0.007 | 0.205±0.008 | 0.048±0.002 | 0.011±0.001 | 6.1 |
| RCut | cardinality | 0.223±0.021 | 0.090±0.010 | 0.208±0.011 | 0.054±0.002 | 0.012±0.002 | 14.9 |
| RCut | train | 0.223±0.021 | 0.090±0.010 | 0.208±0.011 | 0.054±0.002 | 0.012±0.002 | 14.9 |
| RCut | 5-cv | 0.223±0.021 | 0.090±0.010 | 0.208±0.011 | 0.052±0.002 | 0.012±0.002 | 13.3 |
| SCut | train | 0.209±0.024 | 0.093±0.008 | 0.225±0.011 | 0.123±0.017 | 0.095±0.013 | 20.5 |
| SCut | 5-cv | 0.206±0.025 | 0.090±0.009 | 0.200±0.009 | 0.080±0.017 | 0.124±0.035 | 13.1 |
| Meta | regression, content, train | 0.218±0.018 | 0.092±0.010 | 0.213±0.010 | 0.053±0.003 | 0.011±0.002 | 14.1 |
| Meta | regression, scores, train | 0.210±0.024 | 0.092±0.011 | 0.218±0.013 | 0.053±0.003 | 0.011±0.002 | 13.9 |
| Meta | regression, ranks, train | 0.209±0.022 | 0.094±0.009 | 0.211±0.010 | 0.054±0.004 | 0.011±0.001 | 14.5 |
| Meta | regression, content, 5-cv | 0.224±0.018 | 0.090±0.011 | 0.219±0.010 | 0.054±0.003 | 0.011±0.002 | 15.7 |
| Meta | regression, scores, 5-cv | 0.225±0.020 | 0.088±0.010 | 0.214±0.009 | 0.054±0.002 | 0.011±0.002 | 14.6 |
| Meta | regression, ranks, 5-cv | 0.225±0.018 | 0.088±0.010 | 0.213±0.009 | 0.054±0.003 | 0.010±0.002 | 12.3 |
| Meta | classification, content, train | 0.219±0.026 | 0.096±0.011 | 0.226±0.011 | 0.053±0.003 | 0.011±0.002 | 17 |
| Meta | classification, scores, train | 0.208±0.022 | 0.092±0.011 | 0.220±0.010 | 0.054±0.003 | 0.011±0.002 | 14.9 |
| Meta | classification, ranks, train | 0.204±0.027 | 0.092±0.010 | 0.212±0.010 | 0.054±0.003 | 0.011±0.001 | 13.7 |
| Meta | classification, content, 5-cv | 0.225±0.026 | 0.094±0.010 | 0.230±0.010 | 0.054±0.003 | 0.011±0.002 | 19.7 |
| Meta | classification, scores, 5-cv | 0.230±0.023 | 0.088±0.010 | 0.228±0.012 | 0.055±0.003 | 0.011±0.002 | 17.5 |
| Meta | classification, ranks, 5-cv | 0.228±0.020 | 0.088±0.010 | 0.229±0.009 | 0.055±0.003 | 0.011±0.002 | 17.5 |
| TP | content, train | 0.207±0.023 | 0.090±0.008 | 0.205±0.011 | 0.050±0.002 | 0.011±0.001 | 7.6 |
| TP | scores, train | 0.205±0.023 | 0.088±0.008 | 0.213±0.011 | 0.050±0.003 | 0.010±0.001 | 6.4 |
| TP | ranks, train | 0.212±0.024 | 0.098±0.009 | 0.211±0.010 | 0.056±0.006 | 0.017±0.009 | 19.3 |
| TP | content, 5-cv | 0.215±0.018 | 0.101±0.007 | 0.210±0.009 | 0.050±0.002 | 0.011±0.002 | 14 |
| TP | scores, 5-cv | 0.207±0.019 | 0.098±0.005 | 0.207±0.010 | 0.048±0.003 | 0.010±0.002 | 8.2 |
| TP | ranks, 5-cv | 0.212±0.011 | 0.102±0.007 | 0.207±0.009 | 0.049±0.003 | 0.019±0.019 | 14.8 |

TABLE X
HAMMING LOSS OF CLR UNDER DIFFERENT THRESHOLDING STRATEGIES IN ALL DATASETS.

| strategy | parameters | emotions | scene | yeast | enron | medical | rank |
|---|---|---|---|---|---|---|---|
| default | - | 0.242±0.029 | 0.139±0.009 | 0.220±0.009 | 0.047±0.002 | 0.010±0.001 | 14.6 |
| OT | train | 0.222±0.028 | 0.110±0.010 | 0.205±0.008 | 0.050±0.002 | 0.010±0.001 | 3.6 |
| OT | 5-cv | 0.230±0.032 | 0.110±0.010 | 0.205±0.008 | 0.049±0.002 | 0.010±0.001 | 4.8 |
| RCut | cardinality | 0.240±0.025 | 0.113±0.013 | 0.211±0.010 | 0.052±0.002 | 0.013±0.002 | 15.2 |
| RCut | train | 0.240±0.025 | 0.113±0.013 | 0.211±0.010 | 0.052±0.002 | 0.013±0.002 | 15.2 |
| RCut | 5-cv | 0.240±0.026 | 0.113±0.013 | 0.211±0.010 | 0.051±0.001 | 0.013±0.002 | 13.9 |
| SCut | train | 0.227±0.031 | 0.110±0.010 | 0.233±0.009 | 0.074±0.009 | 0.243±0.029 | 18.2 |
| SCut | 5-cv | 0.218±0.031 | 0.108±0.008 | 0.204±0.008 | 0.058±0.008 | 0.073±0.034 | 10.6 |
| Meta | regression, content, train | 0.237±0.027 | 0.114±0.013 | 0.216±0.009 | 0.052±0.002 | 0.012±0.002 | 14.8 |
| Meta | regression, scores, train | 0.220±0.031 | 0.115±0.013 | 0.220±0.008 | 0.052±0.002 | 0.012±0.001 | 13.8 |
| Meta | regression, ranks, train | 0.224±0.030 | 0.114±0.013 | 0.219±0.008 | 0.053±0.002 | 0.011±0.002 | 13.1 |
| Meta | regression, content, 5-cv | 0.227±0.014 | 0.116±0.012 | 0.218±0.011 | 0.052±0.002 | 0.012±0.001 | 14.7 |
| Meta | regression, scores, 5-cv | 0.235±0.016 | 0.114±0.012 | 0.216±0.011 | 0.052±0.002 | 0.012±0.001 | 14.4 |
| Meta | regression, ranks, 5-cv | 0.234±0.017 | 0.114±0.012 | 0.214±0.011 | 0.053±0.002 | 0.012±0.001 | 15.1 |
| Meta | classification, content, train | 0.239±0.027 | 0.117±0.012 | 0.228±0.011 | 0.052±0.003 | 0.011±0.001 | 17.5 |
| Meta | classification, scores, train | 0.220±0.030 | 0.115±0.013 | 0.221±0.011 | 0.052±0.003 | 0.011±0.001 | 12.5 |
| Meta | classification, ranks, train | 0.223±0.027 | 0.116±0.013 | 0.219±0.007 | 0.052±0.002 | 0.011±0.001 | 12.7 |
| Meta | classification, content, 5-cv | 0.229±0.022 | 0.119±0.012 | 0.231±0.012 | 0.052±0.003 | 0.012±0.001 | 17.8 |
| Meta | classification, scores, 5-cv | 0.231±0.022 | 0.114±0.012 | 0.226±0.009 | 0.054±0.003 | 0.012±0.001 | 17.6 |
| Meta | classification, ranks, 5-cv | 0.237±0.019 | 0.114±0.012 | 0.223±0.010 | 0.055±0.003 | 0.012±0.002 | 18.4 |
| TP | content, train | 0.223±0.035 | 0.110±0.009 | 0.209±0.009 | 0.050±0.002 | 0.011±0.001 | 5.5 |
| TP | scores, train | 0.223±0.028 | 0.113±0.011 | 0.218±0.010 | 0.049±0.002 | 0.011±0.001 | 7.6 |
| TP | ranks, train | 0.238±0.026 | 0.132±0.022 | 0.220±0.010 | 0.051±0.002 | 0.011±0.001 | 15.6 |
| TP | content, 5-cv | 0.242±0.027 | 0.130±0.012 | 0.212±0.010 | 0.051±0.002 | 0.013±0.002 | 17.8 |
| TP | scores, 5-cv | 0.228±0.026 | 0.137±0.017 | 0.208±0.008 | 0.049±0.003 | 0.012±0.002 | 11.5 |
| TP | ranks, 5-cv | 0.237±0.028 | 0.139±0.015 | 0.211±0.009 | 0.050±0.003 | 0.012±0.001 | 14.5 |

Another interesting conclusion, that we reached to when examining the results of BP-MLL, is that one can build a high performance multi-label classifier, by building a high performance multi-label ranker and coupling it with a strong thresholding strategy.

Finally, for most datasets and multi-label learning algorithms the *scores* representation seemed to deliver better results compared to the content or the ranks, when used with instance-based methods.

## VI. CONCLUSIONS AND FUTURE WORK

This paper focused on the issue of obtaining a bipartition from the score vector output by a multi-label learning algorithm. This is an important topic, because bipartitions are necessary in a variety of practical applications on one hand,

and because most multi-label learning algorithms output score vectors on the other. It contributed an extensive critical presentation of existing thresholding strategies, and an interesting presentation of a variety of multi-label learning algorithms with emphasis on their output capabilities and thresholding strategy employed. Finally, it contributed an empirical study of these thresholding strategies, drawing conclusions on how to best set their parameters, on which one works best for each algorithm, and on how important is their contribution towards high quality bipartitions.

To improve the impact of this work, we plan to strengthen the empirical part with more multi-label learning algorithms, more and larger datasets and more thresholding strategies, such as the variations of SCut (FBR.0) and (FBR.1) suggested in [17] and further studied in [18]. In addition we plan to investigate thoroughly the relationships between the thresholding strategies, the ranking quality of the multi-label learners and the obtained bipartitions.

## References

[1] A. McCallum, "Multi-label text classification with a mixture model trained by em," in *Proceedings of the AAAI' 99 Workshop on Text Learning*, 1999.

[2] Y. Schapire, R.E. Singer, "Boostexter: a boosting-based system for text categorization," *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.

[3] 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.

[4] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor, "Kernel-based learning of hierarchical multilabel classification methods," *Journal of Machine Learning Research*, vol. 7, pp. 1601–1626, 2006.

[5] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Hierarchical classification: combining bayes with svm," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 177–184.

[6] M. Boutell, J. Luo, X. Shen, and C. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[7] T. Li and M. Ogihara, "Toward intelligent music information retrieval," *IEEE Transactions on Multimedia*, vol. 8, no. 3, pp. 564–574, 2006.

[8] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multilabel classification of music into emotions," in *Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008), Philadelphia, PA, USA, 2008*, 2008.

[9] A. Dimou, G. Tsoumakas, V. Mezaris, I. Kompatsiaris, and I. Vlahavas, "An empirical study of multi-label learning methods for video annotation," in *Proc. 7th International Workshop on Content-Based Multimedia Indexing, CBMI '09*, Chania, Greece, 2009.

[10] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang, "Correlative multi-label video annotation," in *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*. New York, NY, USA: ACM, 2007, pp. 17–26.

[11] A. Clare and R. King, "Knowledge discovery in multi-label phenotype data," in *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2001)*, Freiburg, Germany, 2001, pp. 42–53.

[12] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Advances in Neural Information Processing Systems 14*, 2002.

[13] H. Blockeel, L. Schietgat, J. Struyf, S. Dzeroski, and A. Clare, "Decision trees for hierarchical multilabel classification: A case study in functional genomics," in *Proc. 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2006*, 2006, pp. 18–29.

[14] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Machine Learning*, vol. 73, no. 2, pp. 185–214, 2008.

[15] G. Pandey, C. L. Myers, and V. Kumar, "Incorporating functional inter-relationships into protein function prediction algorithms," *BMC Bioinformatics*, vol. 10, no. 1, p. 142, 2009.

[16] Y. Zhang, S. Burer, and W. N. Street, "Ensemble pruning via semi-definite programming," *Journal of Machine Learning Research*, vol. 7, pp. 1315–1338, 2006.

[17] Y. Yang, "A study of thresholding strategies for text categorization," in *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2001, pp. 137–145.

[18] R.-E. Fan and C. J. Lin, "A study on threshold selection for multi-label classification," National Taiwan University, Tech. Rep., 2007.

[19] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Bringer, "Label ranking by learning pairwise preferences," *Artificial Intelligence*, 2008.

[20] 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.

[21] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random $k$-labelsets for multi-label classification," *IEEE Transactions on Knowledge and Data Engineering*, 2010, (accepted).

[22] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *Proc. 8th IEEE International Conference on Data Mining (ICDM'08)*, 2008, pp. 995–1000.

[23] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," in *Proc. 20th European Conference on Machine Learning (ECML 2009)*, 2009, pp. 254–269.

[24] 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.

[25] J. Fürnkranz, E. Hüllermeier, E. L. Mencia, and K. Brinker, "Multilabel classification via calibrated label ranking," *Machine Learning*, 2008.

[26] L. Tang, S. Rajan, and V. K. Narayanan, "Large scale multi-label classification via metalabeler," in *WWW '09: Proceedings of the 18th international conference on World wide web*. New York, NY, USA: ACM, 2009, pp. 211–220.

[27] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data (accepted)," in *Data Mining and Knowledge Discovery Handbook*, 2nd ed., O. Maimon and L. Rokach, Eds. Springer, 2009.

[28] G. Tsoumakas and I. Vlahavas, "Random $k$-labelsets: An ensemble method for multilabel classification," in *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*, Warsaw, Poland, September 17-21 2007, pp. 406–417.

[29] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Multilabel text classification for automated tag suggestion," in *Proceedings of the ECML/PKDD 2008 Discovery Challenge*, Antwerp, Belgium, 2008.

[30] Y. Song, L. Zhang, and L. C. Giles, "A sparse gaussian processes classification framework for fast tag suggestions," in *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 93–102.

[31] J. Read, "A pruned problem transformation method for multi-label classification," in *Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008)*, 2008, pp. 143–150.

[32] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[33] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[34] Y. Wang and I. H. Witten, "Induction of model trees for predicting continuous classes," in *Poster papers of the 9th European Conference on Machine Learning*. Springer, 1997.

[35] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.