Multi-label Classification with Meta-labels

Jesse Read
Aalto University and HIIT
jesse.read@aalto.fi

Antti Puurula
University of Waikato
asp12@students.waikato.ac.nz

Albert Bifet
Huawei Noah's Ark Lab
bifet.albert@huawei.com

***Abstract*—The area of multi-label classification has rapidly developed in recent years. It has become widely known that the baseline binary relevance approach can easily be outperformed by methods which learn labels together. A number of methods have grown around the label powerset approach, which models label combinations together as class values in a multi-class problem. We describe the label-powerset-based solutions under a general framework of *meta-labels* and provide some theoretical justification for this framework which has been lacking; explaining how meta-labels essentially allow a random projection into a space where non-linearities can easily be tackled with established linear learning algorithms. The proposed framework enables comparison and combination of related approaches to different multi-label problems. We present a novel model in the framework and evaluate it empirically against several high-performing methods, with respect to predictive performance and scalability, on a number of datasets and evaluation metrics. This deployment obtains competitive accuracy for a fraction of the computation required by the current meta-label methods for multi-label classification.**

I. Introduction

Multi-label classification deals with prediction of sets of label indicators for instances, whereas in the more common multi-class and binary-label prediction a single class variable is predicted for each instance. The widely-known *binary-relevance* method (BR) learns the relevance of each label separately as separate binary problems; however this method has been widely criticized [1], [2], [3], [4], [5] because it does not model dependence among labels. Another typical solution is the *label-powerset* (LP) method [1], [2], [5], where each distinct labelset assignment is treated as a class in a multi-class problem, thus modeling labels together and achieving better predictive performance. However, because label combinations are exponential with the number of labels, strategies must be taken to keep the method tractable. Besides this, many labelsets only occur once, or very infrequently, leading to an unbalanced problem that is hard to learn from.

In this paper we unite scalable LP-strategies under a general framework of *meta-labels*. To help motivate the problem, consider Tab. I, a toy example. BR would create 3 binary classifiers, each taking a values $\in \{0,1\}$, and thus missing the mutual exclusive relation between labels A and B. LP creates one multi-class classifier, which assigns values $\in \{010, 101, 011\}$. Thus the relation between A and B is captured. However, one of these combinations (011) only occurs once in the training data. This means a separate decision boundary to learn with only a single example to learn from. One strategy might be to eliminate this instance prior to training, or to learn labels A, B together, and B, C together. In the following we formalize the methodology behind these possible strategies.

TABLE I: A toy example dataset of 7 examples, and 3 labels, with label set $\mathcal{L} = \{A, B, C\}$.

instance	labels	vector
1	B	[0,1,0]
2	B	[0,1,0]
3	B	[0,1,0]
4	A,C	[1,0,1]
5	A,C	[1,0,1]
6	A,C	[1,0,1]
7	B,C	[0,1,1]

TABLE II: Notation and terminology. We index the instance space with $d = 1, \ldots, D$, the label space with $j = 1, \ldots, L$ and the meta-labels space with $k = 1, \ldots, K$.

symbol	description	example
$\mathcal{L}$	label set / L symbols	$\mathcal{L} = \{A, B, C\}$ $(L = 3)$
$y_j \in \{0,1\}$	label relevance	$y_1 = 0$, i.e., A is *not* relevant
$\mathbf{y} = [y_1, \ldots, y_L]$	label assignment	$\mathbf{y} = [0,1,1]$, i.e., labels B, C
$\mathcal{L} = S_1 \cup \cdots \cup S_K$	partition	$\mathcal{L} = \{A, B\} \cup \{B, C\}$
$S_k \subseteq \mathcal{L}$	k-th partition set	$S_1 = \{A, B\}$
$V_k \subseteq 2^{\lvert S_k \rvert}$	possible values for y'_k	$V_1 = \{\emptyset, A, AB\}$
		$\ldots \equiv \{00, 01, 11\}$
		$\ldots \equiv \{0, 1, 3\}$
$y'_k \in V_k$	k-th meta-value	$y'_1 = AB$
$\mathbf{x} = [x_1, \ldots, x_D]$	instance	$\mathbf{x} = [1, 4, 8.8, 0]$
$(\mathbf{x}_i, \mathbf{y}_i)$	i-th training example	See Tab. I

The general multi-label task is to learn from examples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ to produce a classifier that can give predictions $\hat{\mathbf{y}} = h(\tilde{\mathbf{x}})$ for any test instance $\tilde{\mathbf{x}}$. A meta-label[1] represents a combination of labels, e.g., the k-th meta-label might represent combinations of labels A and B, taking values $\mathbf{y}_k \in \{\emptyset, A, B, AB\} \equiv \{00, 01, 10, 11\} \equiv \{0, 1, 2, 3\}$. Tab. II outlines the notation we use. Note that, of course, the actual concepts labels represent depend on the problem domain. For example, in image classification, perhaps $\mathcal{L} = \{\texttt{beach}, \texttt{sunset}, \texttt{foliage}\}$.

Meta-label methods can be carried out in a sequence of three operations:

1) *Partitioning*: partition the label set $\mathcal{L}$ into meta-labels $S_1, \ldots, S_K$, e.g., $\{A, B\} \cup \{C\}$ or $\{A, B\} \cup \{B, C\}$
2) *Relabeling*: decide which values these meta-labels can take, e.g., $y'_1 \in \{01, 10\}$ or $y'_1 \in \{00, 01, 10, 11\}$
3) *Recombination*: recombine meta-label predictions (or confidence scores) into a label vector prediction

Fig. 1 shows example partitions; both overlapping and non-overlapping partition. We can formulate the propagation from

[1]Terminology varies: [6] refers to 'super classes', [1] simply refers to label subsets, upon which *any* multi-label classifier can be trained

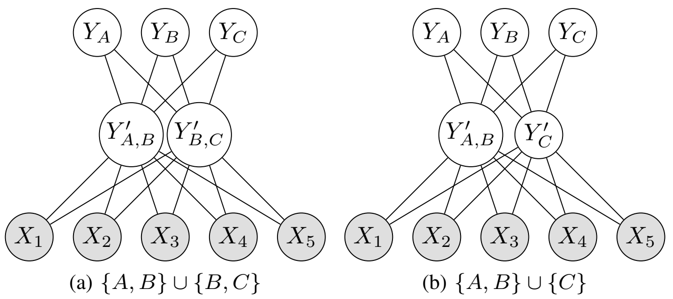


Fig. 1: An example graphical model with meta-labels Y' for the labels Y and instance attributes X. $\mathcal{L} = \{A, B, C\}$ ($L = 3$); $Y_{A,B}$ indicates a meta-label of labels A and B.

the instance space $\mathbf{x}$ to the middle ('hidden') meta layer as

$$y'_k = h_k(\mathbf{x})$$

If meta labels are all binary, this is just a binary relevance approach into the meta-label space, where h_k is the independent model for the k-th meta-label. A second layer propagates from the meta-labels to the labels, and takes a similar form,

$$y_j = g_j(\mathbf{y}') = g_j([h_1(\mathbf{x}), \ldots, h_K(\mathbf{x})])$$

We can easily imagine that, for example,

$$h_k(\mathbf{x}) = \sigma\big(\sum_{d=1}^{D} w_{k,d} x_d\big) \tag{1}$$

and it becomes a neural network, but this is just one possibility. Unlike a typical hidden-layer neural network, however, we can have a deterministic decoding of $\mathbf{y} \leftarrow g(\mathbf{y}')$, and thus back-propagation is not necessary. For example, in Fig. 1a, we can set weights (i.e., links) $w_C = w_{C,C} = w_{B,BC} = w_{A,BC} = 1$ and all other weights to 0; and the Eq. 1 form can be used equally for $g_j(\mathbf{x})$. This means that we only have to train a linear layer, and still have the benefits (non-linearity) of a hidden layer.

Although the concept of 'meta-labels' has been in the multi-label literature for some time (under various terminology), theoretical justification is not always given other than that is advantageous to 'model labels together' or model 'label relationships'.

The meta layer allows us to work in a higher meta space, and allows the application of standard off-the-shelf multi-class classifiers apt to working in such a space. This meta-labels space, is easy to create, yet offers the larger hypothesis space and enables more powerful decision making.

In the following section Section II we elaborate each essential component of the framework; in Section III we describe the ensemble component; in Section IV we review related work; in Section V we carry out empirical evaluations; and in Section VI we make some concluding remarks.

To the best of our knowledge, we are the first to present a unified view of multi-label classification learning with meta-labels, uniting all the diffuse label-powerset-based solutions together. Under this framework we obtain very competitive performance and scalability. In fact, our framework has already contributed to our first and second-place wins on Kaggle competitions [7], [8]

II. A Meta-label Framework

The general process for multi-label classification with meta-labels involves the three steps outlined in Section I plus a choice of multi-class base classifier. The best parameterization ultimately depends on the underlying base classifier and its complexity with respect to the dataset dimensions and properties, and the computational resources available. Later (Section V) we look into the affect that various configurations have on predictive and time performance, and memory use.

A. Partitioning

The main challenge with finding a partition is the sheer number of possibilities to consider. For $L = 6$, there are 115 possible non-overlapping partitions, and for $L = 10$ already $115,975$.

If a hierarchy among labels is defined in the dataset, it is straightforward to partition based on hierarchical proximity, as in, e.g., [9]. In the absence of a predefined hierarchy, one can partition on several strategies. HOMER [5] forms a hierarchy based on balanced k-means clustering of the instance space. [6] finds a good partition with simulated annealing search of conditional label dependence, extended from the work of [4] (marginal label dependence only).

Partitioning heuristically involves some cost; the method in [6] did not scale well to large datasets, and although that of [4] was faster, there is little evidence to indicate that marginal label dependence (i.e., ignoring the input space) is any better than random partitions. It was specifically demonstrated that a random partition can work just as well as a dataset-defined partition [9]. In fact, the RAkEL method [1] demonstrated the effectiveness of a random overlapping partition of meta-labels.

B. Relabeling

Suppose that we partition into $\mathcal{L} = \{A, B\} \cup \{B, C\}$ (as in Fig. 1a). A common formulation (e.g., in [1], [6]), is to form meta-labels from the partition, as for example

$$Y'_{A,B} \in \{00, 01, 10, 11\}$$

The basic LP approach can be applied here – simply by employing a multi-class classifier.

However, if $Y_{AB} = 11$ (labels A and B together) is only relevant to a single (or very few) instance in the dataset, it can impede learning. The Pruned Sets (PS) method [2] prunes away p-infrequent partitions (those which don't occur at least p times). If $p = 1$, then we now have

$$Y_{AB} \in \{00, 10, 01\}$$

As before, standard LP (a multi-class classifier) can now be applied to learn each meta-label. PS was presented as an alternative to subset/partitioning methods like RAkEL, being more efficient than LP by this pruning mechanism, rather than partitioning the label set into subsets. Actually – and apparently not noticed earlier in the literature – the two can be combined easily in the same framework as we present later in Section V; partitions (i.e., meta-labels) can be pruned.

As noticed in [2], if many combinations occur only once, we may end up discarding much of the data. To combat this problem, instances can be reintroduced with subsets. Instead of discarding the instance with the unique combination A, B, PS can make two new instances, and give one of them A (10) and the other one B (01). Namely, in [2], the 'top n' sets are reintegrated (or as many exist up to a maximum of n), where sets are ranked by size and then frequency, e.g., for $n = 2$, a pruned instance of labels A, B, C may be reintegrated as two instances – one labeled AB and the other BC – supposing that there is no bigger combination than ABC in the training data, and that AB and BC are both more frequent than AC.

Note the special case, when we prune all the way down to two combinations, we end up with binary meta-labels, such as

$$Y_{AB} \in \{0, 1\} \equiv \{00, 11\}$$

(we either assign both labels A and B or neither of them). This special case provides particular advantages: it is conceptually simpler, makes learning faster, and reconstruction of label vectors (as we address in Section II-C) is easier. It can also be derived without consideration of 'partitioning' at all. For example, itemsets A, B, AB, ABC, BC can be obtained from the dataset, and used directly. Due to their binary nature, we can generate many and incur the same computational cost as the multi-class meta-label methods. Although, this is exactly the disadvantage: many more must be generated to get the same decision power, and the generation process must take care to cover all labels appropriately (the top n itemsets may not contain certain labels).

C. Recombination

If the partition is disjoint/non-overlapping, recombination of meta-labels y'_k into labels y_j is trivial,

$$\hat{y}_j = g_j(\mathbf{y}') := \sum_{k=1}^{K} \big[j \in y'_k\big] \tag{2}$$

where $y'_k \in V_k$ and $\big[j \in y'_k\big] = 1$ if y'_k's value indicates j's relevance (e.g., if $y'_k = AB$, then $A \in y'_k$ and $C \notin y'_k$); and $\hat{y}_j \in \{0, 1\}$. For example, $\mathbf{y}' = [y'_{AB}, y'_C] = [01, 1]$ recombines to $\mathbf{y} = [y_A, y_B, y_C] = [0, 1, 1]$. For overlapping partitions, the sum for a label may exceed 1 and thus a threshold is necessary,

$$\hat{y}_j = g_j(\mathbf{y}') = \big[p(y_j) > t\big] \tag{3}$$

$$= \Big[\sum_{k=1}^{K} \mathbf{P}_{jk} > t\Big] \tag{4}$$

for some threshold t, where $\mathbf{P}$ is a $L \times K$ matrix of the posterior probability for each label j given the value of each meta label k;

$$\mathbf{P}_{jk} = \begin{cases} 0 & \text{if } j \notin V_k \\ \sum_{v \in V_k} \big[j \in V_k\big] \cdot p(y'_k = v|\mathbf{x}) & \text{otherwise} \end{cases}$$

where $p(y'_k = V|\mathbf{x})$ is the posterior given by $h(\mathbf{x})$. An example is given in Tab. III.

TABLE III: Given a meta-label $y'_k \in \{00, 01, 11\}$ for $S_k = \{A, B\}$, and a posterior predictive distribution $p(y'_k = v|\tilde{\mathbf{x}})$ for $h_k(\mathbf{x})$, the degree of relevance of the labels A and B to $\tilde{\mathbf{x}}$ can be approximated by distribution $\mathbf{P}_{\cdot k}$.

j:	A	B	$p(y'_k = v\|\mathbf{x})$
v_1	0	0	0.0
v_2	0	1	0.9
v_3	1	1	0.1
$\mathbf{P}_{\cdot k}$	0.1	1.0	

TABLE IV: Examples of recombination of meta-label votes into a label vector. In most cases, we obtain votes upon which we can apply a thresholded to obtain predictions $\hat{y}_j \in \{0, 1\}$ for $j = 1, \ldots, L$, as per Eq. 3.

(a) Disjoint, Eq. 2

$\mathcal{L}$	A	B	C
$y'_{A,B}$	0	1	
y'_C			0
$\hat{\mathbf{y}}$	0	1	0

(b) With confidence outputs, Eq. 3, Eq. 4

$\mathcal{L}$	A	B	C
$\mathbf{P}_{jk}$	0.1	1.0	
$\mathbf{P}_{jk}$		0.7	0.3
$p(y'_j) = \sum_k \mathbf{P}_{jk}$	0.1	1.7	0.3
$\hat{\mathbf{y}} = \big[\sum_k \mathbf{P}_{jk} > 0.5\big]$	0	1	0

Tab. IV steps through the different recombination possibilities. Given appropriate choices for weights $w_{k,j}$ all these schemes can be represented as

$$y_j = g_j(\mathbf{y}') = \sum_{k=1}^{K} h_k(\mathbf{x}) \cdot w_{k,j}$$

where $w_{k,j}$ indicates the contribution of each k-th meta-label to the estimated relevance of each j-th label; as if it were the hidden layer of a standard neural network scheme.

Because any binary digit can be encoded as decimal, both the multi-label and multi-target (i.e., each label is multi-valued) cases are equally applicable in this framework, both in the cases of the labels and the meta labels. For example, with binary meta labels $Y_A \in \{0, 1\}, Y_B \in \{0, 1\}, Y_{AB} \in \{00, 11\}\}$, we obtain identical results as in Fig. III (which had meta label $Y_{AB} \in \{01, 10, 11\}$) when $p(y'_A = 1) = 0.0$ and $p(y'_B = 0) = 1.0$ and $p(y'_{AB} = 1) = 0.1$ (in Eq. 3).

III. Ensembles

Ensemble methods have gained and retained popularity in the multi-label literature [1], [2], [6] (as well as the machine learning literature in general) in spite of their inherent tendency to use more memory and computation power. Meta-labels offer a high-dimensional class space and can induce a powerful model but also high variance across different training sets. Ensembles are a good way to deal with this.

The ensemble strategy is in fact already used within several 'standalone' methods. For example, RAkEL is sometimes referred to as en ensemble scheme of M models, however these models are not standalone, and only model a subset of the labels. We note that several RAkELs could be used together in ensemble, as with any other multi-label method.

Given output $\hat{\mathbf{y}}^{(m)} = [\hat{y}_1, \ldots, \hat{y}_L]$, from a multi-label classifiers $m = 1, \ldots, M$, then, for example,

$$\hat{y}_j = f_j([\hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(M)}] \approx [\frac{1}{M}\sum_{m=1}^{M} \hat{y}_j^{(m)} > t]$$

Where the end term is just a simple majority vote ensemble with a threshold. More advanced schemes (which we do not cover here) are equally applicable. This can in fact be seen as an additional layer on the network, such that

$$\hat{y}_j = f_j(g(h(\mathbf{x})))$$

and all layers can be represented as a four-layer neural network,

$$\hat{z}_j^{\ell} = h\Big(\sum_{k=1}^{K} h_k(\mathbf{z}^{\ell-1}) \cdot w_{k,j}\Big) \tag{5}$$

for layers $\ell = 1, \ldots, 4$, and where $h_k(z^{\ell-1})$ computes a function on the input of the previous layer ℓ, and $w_{k,j}$ is some weight $\mathbf{z}^1 = \mathbf{x}$ is the first layer, and $\mathbf{z}^4 = \mathbf{y}$ is the final layer after ensemble voting. This formulation as neural networks provides a theoretical base to lean on, however, we emphasise that any linear classifier can be plugged into layer one and the weights of all remaining layers, while adding depth and predictive power, are deterministically encodable.

IV. Related Work

The most closely related work is that of RAkEL [1] and HOMER [5] which we already described above and compare to in the empirical evaluation of the following section.

Creating meta-labels to represent label combinations can be viewed as compressing the label space. Compressed sensing addresses the problem in a related way, by predicting compressed labels [10], taking advantage of the fact that, for sparse labeling, the number of subproblems are logarithmic with the total number of possible labels. PCA can compress the label space into a different dimension [11]. Again here, prediction is done in this new dimension, and then predictions are cast back into the original label space.

We have already mentioned connections to neural-networks. BPMLL [3] is a multi-label neural network approach well known in the multi-label literature, which trains a hidden layer and uses backpropagation. Our framework does not require backpropagation, because the hidden layer can be seen as a random projection from the label space. Extreme learning machines [12] (ELMs) are based on a similar principal (but not designed specifically for multi-label classification), and can be seen as multi-layered neural networks where the middle layer comes from random feature functions, making the network shallower (in terms of which parameters have to be learned) and thus easier to learn. A difference of ELMs is that the middle hidden layer is projected forward from the input space, whereas we project backward from the label space (which is available to us in the multi-label context).

V. Experiments

We implemented our meta-label approaches using the Meka framework[2] and our source code is available in the most recent version. As a base classifier to learn meta-labels, we use support vector machines (SVMs) – namely, Weka's SMO implementation – on account of its popularity and good out-of-the-box performance, as found elsewhere in the literature ([1], [2], [13]). We display results of

$$\textsc{Accuracy} := \frac{1}{N}\sum_{i=1}^{N} \frac{|\mathbf{y}_i \wedge \hat{\mathbf{y}}_i|}{|\mathbf{y}_i \vee \hat{\mathbf{y}}_i|}$$

as Accuracy in, e.g., [2], [13], known commonly as Jaccard Index in information retrieval. We obtained similar relative results under other measures (not displayed due to lack of space) which can be confirmed by using our implementations on the Meka framework.

TABLE V: Multi-label / Meta-label Methods

Key	Name / Description
BR	Binary Relevance (L partitions)
LP	Label Powerset (1 partition)
Hd	Hierarchical [dataset-defined] disjoint partitions (with LP)
Rd	K Random disjoint equal sized partitions (with LP)
RAkELp	Random overlapping k-sized *pruned* partitions (with pruned LP)
HOMER	K Hierarchical [random equalsized, via clustering] partitions (with LP)
BPMLL	Backprop. neural network
EpRd	Ensembles of 10 *pruned* Rd (with pruned LP)

The main methods we look at are listed in Tab. V, although we describe particular variations as we present results. We do not normally consider BR and LP meta-label methods because they do not have a separate 'middle' layer of labels – but they can be viewed as two 'extremes' of meta-labeling: LP has a single meta-label of up to 2^L values, and BR has L meta-labels of only 2 values ($\in \{0, 1\}$), which correspond to the original labels. Although the components behind them already existed in the literature, RAkELp and EpRd are novel combinations of those components which we present within our framework.

Tab. VI lists the datasets we consider. All datasets in this collection have been studied in a large scale comparison [13], and are available for download[3].

TABLE VI: Datasets and associated statistics; LC is *label cardinality* (average number of labels relevant to each example).

	N	L	d	LC	Type
emotions	593	6	72	1.87	audio
scene	2407	6	294	1.07	image
yeast	2417	14	103	4.24	biology
medical	978	45	1449	1.25	medical/text
enron	1702	53	1001	3.38	text
tmc2007	28596	22	500	2.16	text
mediamill	43907	101	120	4.38	video
bibtex	7395	159	1836	2.40	text
delicious	16105	983	500	19.02	text
bookmarks	87856	208	2150	2.28	text

Before beginning a general evaluation, we analyzed a number of methods on the enron dataset. This dataset has a reasonable number of labels (53) and, more importantly, a predefined hierarchy[4], and thus allows us to compare the

[2]http://meka.sourceforge.net

[3]http://mulan.sourceforge.net/datasets.html and http://sourceforge.net/projects/meka/files/Datasets/Train-testSplits/

[4]See http://bailando.sims.berkeley.edu/enron/enron_categories.txt

TABLE VII: Methods' performance on the enron dataset. The dataset's predefined hierarchy has 4 leaves, resulting in 4 relatively-balanced partitions. Thus we also set $K = 4$ for HOMER, Rd. BR and LP can be seen as having $K = L$ and $K = 1$, respectively. Note that we have taken an average of Rd's performance (since partitions are random) with different random seeds.

Method	Accuracy	Time (s)
BR	0.406	25
LP	0.434	426
HOMER	0.436	93
Hd	0.428	37
Rd	0.435±0.006	121±6

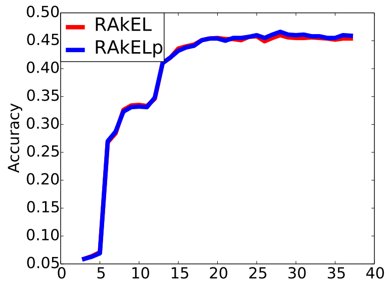
various partitioning methods to form meta-labels including that of using the dataset-defined hierarchy. Results are in Tab. VII.

As expected, LP is least scalable and BR performs poorest. Partitioning along a dataset defined hierarchy appears to offer no better performance than a random partition. This should not be too disappointing since often there is no such hierarchy to use anyway. Surprisingly, HOMER offered relatively little advantage over either a random or dataset-defined hierarchy partition. These results indicated to us to proceed along the line of the Rd method.

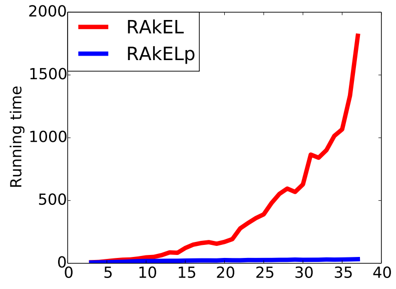
RAkEL is a special case of meta-labels with overlapping partitions, rather than disjoint partitions like Rd (namely it has M partitions randomly drawn from $\mathcal{L}$ of size k [5], i.e., $S_1, \ldots, S_K$ where all $|S_k| = \mathsf{k}$. In Fig. 2, we fix $M = 10$, and increase k. We see that higher k offers higher accuracy (a result already reported in [1]), but as also noticed in [1], processing time quickly becomes unattractive, and scaling up to larger datasets with the same parameters is not feasible. In our general meta-labels framework, we prune the number of class values *within* each of RAkEL's partitions (RAkELp; the original RAkEL method does no pruning) but also reintegrate pruned instances, as described in Section II-B, see also [2]. With pruning, time complexity is almost constant and certainly not more than linear (wrt k), yet the dimension offered by this meta-label space still allows for high predictive performance. In fact, predictive performance is practically indistinguishable from RAkEL (if anything, slightly higher), whereas time *never exceeds* by more than a few seconds that of BR (25 seconds, see Tab. VII – wheras RAkELp takes at most 32 seconds at $k = 37$).

For ensembles of Rd (i.e., M ensemble members, each with a disjoint split into K meta-labels, thus $|S_k| = \frac{L}{K}$). Memory complexity is $M \times (D + k)$ and time complexity is $T \times N \times D \times \min(N, 2^k)$, where T approximates the complexity of the base learning (e.g., in the case of gradient descent, T is the maximum number of learning iterations carried out). The base learner can easily be changed (thus reducing T), however there can be considerable difference in accuracy given different base learners, which is why top-performing problem-transformation methods use support vector machines (e.g., [14], [2], [13]), rather than something faster like naive Bayes). Therefore we are often (understandably) reluctant to

[5]RAkEL's k is not the same as our index variable k for meta-labels



(a) Accuracy under increasing k



(b) Running time (s) under increasing k

Fig. 2: Performance on Enron with M overlapping partitions of size k (i.e., RAkEL) and with meta-labels pruned at $p = 5, n = 1$ (See Section II-B; examples discarded if meta-label value occurs less than 5 times, but potentially reintegrated with a more frequent labelset).

use a poorer-performing learner. Another solution to reduce complexity is to subsample the input space (in terms of N and D) for each model, as done in [15], among others, where they reported similar accuracy for as low as 50% of the original input data per model. This strategy is not specific to meta-labels, rather can easily be added to any ensemble framework (a specific case, where decision trees are involved, is a called a random forest). The remaining terms are M and 2^k. Using disjoint partitions means that each partitioning is a complete model (unlike, say, RAkEL) and thus we can keep M quite low (specifically, we use 10). Lastly, by using pruning, it is straightforward to reduce the 2^k term to a constant $c \ll 2^k$ – determined by an appropriate choice for p, i.e., we can prune the values of a particular meta-label until we get to a desired c values.

We next took an appropriate configuration of the meta-label framework (EpRd: Ensembles of meta-labels based on Random Disjoint partitions, with Pruning), guided by our initial analysis, and compared it to some existing configurations from the literature. We chose parameters to yield high scalability along with good predictive performance: $M = 10, K = \frac{L}{10}+1$ for all datasets, and pruning of $p = 5, n = 1$ for datasets emotions to enron (top to bottom in Tab. VI), $p = 7, n = 1$ up till bibtex, $p = 20, n = 0$ for delicious and $p = 100, n = 0$ for bookmarks. We use the same dataset splits as in [13] and thus our results are directly comparable, taking into account the following: [13] uses the libsvm implementation as a base classifier, whereas we use Weka's SMO implementa-

TABLE VIII: General results.

Accuracy (Jaccard Index)

Dataset	BR	HOMER	RAkEL	BPMLL	EpRd
emotions	0.361	0.471	0.419	0.517	**0.542**
scene	0.689	0.717	**0.734**	0.653	0.724
yeast	0.520	**0.559**	0.531	0.525	0.541
medical	0.206	0.713	0.673	0.495	**0.759**
enron	0.446	**0.478**	0.428	0.343	0.473
corel5k	0.030	**0.179**	0.000	0.133	0.131
tmc2007	0.891	**0.888**	0.852	0.540	0.635
mediamill	0.403	0.413	0.337	0.373	**0.420**
bibtex	0.348	**0.330**	DNF	0.242	0.233
delicious	0.136	**0.207**	DNF	0.159	0.168
bookmarks	DNF	DNF	DNF	0.147	**0.181**

Training time

Dataset	BR	HOMER	RAkEL	BPMLL	EpRd
emotions	4	4	5	**1**	3
scene	71	68	79	6	**5**
yeast	145	101	157	**6**	**15**
medical	18	16	82	15	4
enron	318	158	493	**20**	36
corel5k	926	771	3380	**191**	952
tmc2007	42 645	31 300	102 394	**180**	3 819
mediamill	85 468	78 195	33 554	**369**	19 221
bibtex	11 013	2896	DNF	**177**	230
delicious	57 053	21218	DNF	**3 648**	51 578
bookmarks	DNF	DNF	DNF	**2 682**	9 290

tion; We used 3.20 GHz processors and up to 2 GB RAM, whereas [13] used 2.50 GHz and 64 GB. Additionally to baseline BR, RAkEL, and HOMER from [13], we deployed the backpropagation neural network BPMLL from [3] with the parameters suggested in that paper (one hidden layer, 100 training iterations, learning rate of 0.01). We considered this it a fitting comparison due to the similarity in formulation with meta-labels. Needless to say that we did not compare to LP since on most of these datasets it is completely intractable. All methods not finishing in under a week are considered Did Not Finish (DNF).

Tab. VIII displays the results of the general evaluation. Our configuration, under a general framework of meta-labels, obtains overall similar predictive performance as the deployment of other related methods, with considerable reduction in run time (more efficient even than even baseline BR). A standard backprop neural network is comparably fast but cannot offer anywhere near the same predictive power.

On bookmarks, most methods did not finish even after one week, whereas EpRd only took a few hours, and still outperformed the neural-network (BPMLL). On tmc2007 and bibtex results were disappointing for EpRd. It is possible that we set the pruning parameter too high in these cases. We intend to study these cases in future investigation.

VI. Conclusions

We presented a general framework for multi-label learning with meta-labels. Meta-labels are a flexible problem-transformation based approach that has been taken up at various points in the literature, on the basis that they model labels together (unlike binary relevance learning) and can be more efficient than learning all labels together (as in the label powerset method). A variety of meta-label methods exist in the literature, but have not been well connected. We have united the best of these approaches under a single framework, with theoretical backing, and explained how the projection of labels into a new space is the main element behind the predictive performance. Having a general framework, we were able to identify strategies that were previously presented and evaluated individually, and combine them into novel methods. As an example, we combined pruning strategies with meta-label formation from disjoint partitions, within an ensemble scheme. We empirically evaluated this approach and found a dramatic improvement in running time (up to orders of magnitude) related high-performance multi-label learners, with predictive performance at least as good or better than those methods. In future work, we will carry out further empirical evaluation on larger datasets.

Acknowledgment

The first author is funded by the Aalto Energy Efficiency project, http://energyefficiency.aalto.fi/en/.

References

[1] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-labelsets for multi-label classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 99, no. 1, 2010.

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

[3] M.-L. Zhang and Z.-H. Zhou, "Multilabel 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.

[4] L. Tenenboim, L. Rokach, and B. Shapira, "Identification of label dependencies for multi-label classification," in *MLD '10: 2nd International Workshop on Learning from Multi-Label Data from ICML/COLT 2010*, 2010.

[5] G. Tsoumakas, I. Katakis, and I. P. Vlahavas, "Effective and efficient multilabel classification in domains with large number of labels," in *ECML/PKDD 2008 Workshop on Mining Multidimensional Data*, 2008.

[6] J. Read, C. Bielza, and P. Larrañaga, "Multi-dimensional classification with super-classes," *Transactions on Knowledge and Data Engineering*, Accepted for publication.

[7] A. Puurula, J. Read, and A. Bifet, "Kaggle LSHTC4 winning solution," http://www.kaggle.com/c/lshtc, Tech. Rep., 2014.

[8] G. Tsoumakas, A. Papadopoulos, W. Qian, S. Vologiannidis, A. D'yakonov, A. Puurula, J. Read, J. Svec, and S. Semenov, "WISE 2014 challenge: Multi-label classification of print media articles to topics," in *Web Information Systems Engineering - WISE 2014, Proceedings, Part II*, 2014, pp. 541–548.

[9] J. Read, "Scalable multi-label classification," Ph.D. dissertation, University of Waikato, 2010.

[10] D. Hsu, S. M. Kakade, J. Langford, and T. Zhang, "Multi-label prediction via compressed sensing," in *NIPS '09: Neural Information Processing Systems 2009*, 2009.

[11] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik, "Kernel dependency estimation," in *NIPS*, 2002, pp. 873–880.

[12] G.-B. Huang, D. Wang, and Y. Lan, "Extreme learning machines: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011. http://dx.doi.org/10.1007/s13042-011-0019-y

[13] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recognition*, vol. 45, no. 9, pp. 3084–3104, Sep. 2012.

[14] G. Tsoumakas and I. P. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," in *ECML '07: 18th European Conference on Machine Learning*. Springer, 2007, pp. 406–417.

[15] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning*, vol. 85, no. 3, pp. 333–359, 2011.