A Literature Survey on Algorithms for Multi-label Learning

Mohammad Sorower
Philips

A Literature Survey on Algorithms for Multi-label Learning

Mohammad S Sorower
Department of Computer Science
Oregon State University
Corvallis, OR 97330
sorower@eecs.oregonstate.edu

Abstract

Multi-label Learning is a form of supervised learning where the classification algorithm is required to learn from a set of instances, each instance can belong to multiple classes and so after be able to *predict* a set of class labels for a new instance. This is a generalized version of most popular *multi-class* problems where each instances is restricted to have only one class label. There exists a wide range of applications for *multi-labelled* predictions, such as text categorization, semantic image labeling, gene functionality classification etc. and the scope and interest is increasing with modern applications. This survey paper introduces the task of *multi-label* prediction (classification), presents the sparse literature in this area in an organized manner, discusses different evaluation metrics and performs a comparative analysis of the existing algorithms. This paper also relates *multi-label* problems with similar but different problems that are often reduced to *multi-label* problems to have access to wide range of *multi-label* algorithms.

1 Introduction

In machine learning, *single label* classification is a common learning problem where the goal is to learn from a set of instances, each associated with a unique class label from a set of disjoint class labels $\mathcal{L}$. Depending on the total number of disjoint classes in $\mathcal{L}$, the problem can be identified as *binary* classification (when $|\mathcal{L}| = 2$) or *multi-class* classification (when $|\mathcal{L}| > 2$) problem. Unlike these problems, *multi-label* classification allows the instances to be associated with more than one class. That is, the goal in multi-label classification is to learn from a set of instances where each instance belong to one or more classes in $\mathcal{L}$.

Even though *multi-label* classification was primarily motivated by the emerging need for automatic text-categorization and medical diagnosis, recent realization of the omnipresence of multi-label prediction tasks in real world problems drawn more and more research attention to this domain [55]. For example, a text document that talks about scientific contributions in medical science can belong to both *science* and *health* category, genes may have multiple functionalities (e.g. diseases) causing them to be associated with multiple classes, an image that captures a field and fall colored trees can belong to both *field* and *fall foliage* categories, a movie can simultaneously belong to *action, crime, thriller*, and *drama* categories, an email message can be tagged as both *work* and *research project*; such examples are numerous. Traditional *binary* and *multi-class* problems both can be posed as specific cases of *multi-label* problem. However, the generality of *multi-label* problems makes it more difficult than the others [66].

This survey paper aims at: i) a structured summarization of different multi-label classification approaches, ii) a systematic presentation of the evaluation measures, and iii) a short summarization

Table 1: Symbols and Notations Used in this paper

Definition	Symbol/Notation
Instance Space	$\mathcal{X}$
Label Set	$\mathcal{L}$
Instance	$\mathbf{x} = (x_1, x_2, ..., x_m) \in \mathcal{X}$
Number of Features	m
Number of Labels	$k = \lvert\mathcal{L}\rvert$
Set of relevant labels (for an instance)	L
Correct Label Vector	$\mathbf{Y} = (y_1, y_2....., y_k)$, $y_i \in \{0, 1\}$, $1 \leq i \leq k$
Label Vector Space	$\mathcal{Y} = \{0, 1\}^k$
Correct Label Set	$\mathbf{Y}^l = (y^l_1, y^l_2....., y^l_p)$, $y^l_i \in \mathcal{L}$, $1 \leq i \leq p$
Label Presence	$Y^\lambda \in \{0, 1\}$
Number of labels for an instance	p
Training dataset	$S = (\mathbf{x}_i, Y_i)$, $1 \leq i \leq n$
Number of Instances	n
Classifier predictions	$\mathbf{Z} = (z_1, z_2....., z_k)$, $z_i \in \{0, 1\}$, $1 \leq i \leq k$
Predicted Label Set	$\mathbf{Z}^l = (z^l_1, z^l_2....., z^l_p)$, $z^l_i \in \mathcal{L}$, $1 \leq i \leq p$
Rank of a label	$r(\lambda)$
Set of classifiers	H
Classifier	h

of benchmark multi-label datasets and finally, iv) elementary performance comparisons of different multi-label algorithms.

Multi-label Learning and Fuzzy Logic Based Learning It is important to distinguish between multi-label classification and fuzzy logic based classification. While they both define membership functions to deal with multiple classes, but the goal and the problems addressed are quite different [3]. Fuzzy logic often deals with the ambiguity in the feature space and used as an additional block (e.g. to find weights the features) before classification to help distinguishing between multiple classes. In contrast, multi-label classification is about labeling an instance with one or more classes. Fuzzy based classification is often followed by a de-fuzzification step that makes the final classification decision. Fuzzy membership values when normalized usually sum up to 1 where there is no such requirement for multi-label classification; even each class in multi-label classification can have a membership value of 1 (ideally). Usually, fuzzy membership values over different classes are correlated but membership values for multi-label classification could be just coincidence.

Notations Let $\mathcal{X}$ be an instance space, and $\mathcal{L} = \{\lambda_1, \lambda_2, ..., \lambda_k\}$ be a finite set of class labels. An instance (or an example) $\mathbf{x} \in \mathcal{X}$, represented in terms of features vector $\mathbf{x} = (x_1, x_2, ..., x_m)$, is (non-deterministically) is associated with a subset of labels $L \in 2^{\mathcal{L}}$. Notice that if we call this set L be the set of relevant labels of **x**, then we could call the complement $\mathcal{L} \setminus L$ to be the set of irrelevant labels of x. Lets denote the set of relevant labels L with a binary vector $\mathbf{Y} = (y_1, y_2....., y_k)$, where $y_i = 1 \iff \lambda_i \in L$. $\mathcal{Y} = \{0, 1\}^k$ is the set of all such possible labelings. Table 1 summarizes all symbols and notations used in this paper. In some cases (e.g. k means, topic model) slightly different notation has been used to be uniform with common practice in the literature.

The rest of this paper is organized as follows: the next section introduces formal definition of multi-label learning and and a few but quite different methods for multi-label learning. Then the next two sections introduce two important issues: exploiting label structure and scaling with large data; followed by a section that describes the challenges with online multi-label classification and a few emerging researches. Section 6 describes a few related problems that are are often reduced to a multi-label problem to obtain access to whole set of multi-label algorithms. Section 7 introduces a set of evaluation metrics and section 8 describes a few benchmark datasets. Finally, performance comparison is followed by future possibilities and concluding remarks.

Table 2: Example Multi-label dataset

Instance	Label Set
1	$\{\lambda_2, \lambda_3\}$
2	$\{\lambda_1\}$
3	$\{\lambda_1, \lambda_2, \lambda_3\}$
4	$\{\lambda_2, \lambda_4\}$

2 Multi-label Learning

Definition 2.1 *Multi-label Learning (MLL):*

Given a training set, $S = (\boldsymbol{x}_i, \boldsymbol{Y}_i)$, $1 \leq i \leq n$, consisting n training instances, $(\boldsymbol{x}_i \in \mathcal{X}, \boldsymbol{Y}_i \in \mathcal{Y})$ i.i.d[1] *drawn from an unknown distribution D, the goal of the multi-label learning is to produce a multi-label classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ (in other words, $h : \mathcal{X} \rightarrow 2^{\mathcal{L}}$) that optimizes some specific evaluation function (i.e. loss function) [66].*

Often it is common for multi-label classifiers to produce a real valued function, $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}$, instead of a direct classification as defined above. Any such method would require some kind of threshold to finally classify between relevant and irrelevant labels.

2.1 Simple Problem Transformation Methods

There exists a number of very simple problem transformation methods which actually transform multi-label data in such a way so that existing classification algorithms (i.e. binary classifiers) can be applied [55]. It would be easier to describe such algorithms using an example multi-label data in Table 2. Notice that the instances features are ignored here, because they are not really important for describing these algorithms. There are four instances belong to at least one of the four classes, $\lambda_1, \lambda_2, \lambda_3, \lambda_4$.

The *copy transformation* method replaces each example (x_i, Y_i) with $|Y_i|$ examples (x_i, λ_j), for each $\lambda_j \in Y_i$. An extension to this is to use an weight of $\frac{1}{|Y_i|}$ to each of these newly created examples. This is called *dubbed copy-weight* method.

For each instance, the *select* family of transformation methods replaces Y_i by one of its members. Depending on how this *one member* is selected, there can be several versions, namely, *select-min* (select least frequent), *select-max* (select most frequent), and *select-random* (randomly selected).

Finally, *ignore trnasformation* simply ignores the multi-label instances and runs the training with single label instances only.

Table 3: Transformed multi-label data: (a) copy (b) dubbed copy-weight (c) select-max (d) select-min (e) select-random (f) ignore

Ex.	Label
1a	λ_2
1b	λ_3
2	λ_1
3a	λ_1
3b	λ_2
3c	λ_3
4a	λ_2
4b	λ_4

(a)

Ex.	Label	Weight
1a	λ_2	0.5
1b	λ_3	0.5
2	λ_1	1.00
3a	λ_1	0.33
3b	λ_2	0.33
3c	λ_3	0.33
4a	λ_2	0.50
4b	λ_4	0.50

(b)

Ex.	Label
1	λ_2
2	λ_1
3	λ_2
4	λ_2

(c)

Ex.	Label
1	λ_3
2	λ_1
3	λ_1
4	λ_4

(d)

Ex.	Label
1	λ_3
2	λ_1
3	λ_2
4	λ_4

(e)

Ex.	Label
2	λ_1

(f)

Table 3 shows datasets produced by these approaches. Notice that none of these methods is likely to retain the actual data distribution and therefore is likely to have lower prediction performance.

[1] independent and identically distributed

Table 4: Transformed data using Label Powerset method

Instance	Label
1	$\lambda_{2,3}$
2	λ_1
3	$\lambda_{1,2,3}$
4	$\lambda_{2,4}$

Label Powerset (LP) *Label Powerset (LP)* is a straight forward method that considers each unique set of labels in a multi-label training data as one class in the new transformed data. Therefore, the new transformed problem is a single label classification task. For a new instance, LP outputs the most probable class which actually is a set of classes in the original data. It is also possible to produce a ranking of labels using LP, given the classifier can output a probability distribution over all newly formed classes [41]. Table 4 shows the dataset transformed using LP method. Notice that the computational complexity of LP is upper bounded by $min(n, 2^k)$, where n is the total number of data instances and k is the total number of classes in the training data (before transformation). In practice complexity would be much less than 2^k, still for large values of n and k this can be an issue. The second problem with this approach is that, a large number of classes (set of a labels in the original data) would be associated with very few examples and that would also pose extreme class imbalance problem for learning.

The *Pruned Problem Transformation* method [41] addresses the second problem above by pruning away the label sets that occur less than a user-defined threshold and often replacing them by introducing disjoint subsets of these labels sets that are more frequent (greater than the threshold) in the data.

Binary Relevance (BR) *Binary Relevance (BR)* is one of the most popular approaches as a transformation method that actually creates k datasets ($k = |\mathcal{L}|$, total number of classes), each for one class label and trains a classifier on each of these datasets. Each of these datasets contains the same number of instances as the original data, but each dataset $D_{\lambda_j}, 1 \leq j \leq k$ positively labels instances that belong to class λ_j and negative otherwise. Table 5 show the example dataset transformed for BR.

Table 5: Transformed data sets produced by Binary Relevance (BR) method

Ex.	Label
1	$\neg\lambda_1$
2	λ_1
3	λ_1
4	$\neg\lambda_1$

(a)

Ex.	Label
1	λ_2
2	$\neg\lambda_2$
3	λ_2
4	λ_2

(b)

Ex.	Label
1	λ_3
2	$\neg\lambda_3$
3	λ_3
4	$\neg\lambda_3$

(c)

Ex.	Label
1	$\neg\lambda_4$
2	$\neg\lambda_4$
3	$\neg\lambda_4$
4	λ_4

(d)

Once these datasets are ready, it is easy to train one *binary classifier* for each. For any new instance, BR outputs the union of the labels λ_j that are positively predicted by the k classifiers. While BR has been used in many practical applications, it has been widely criticized for its implicit assumption of *label independence* which might not hold in the data.

Ranking by Pairwise Comparison (RPC) *Ranking by Pairwise Comparison (RPC)* [24] transforms the multi-label dataset into $\binom{k}{2}$ binary label datasets, one for each pair of labels, $(\lambda_i, \lambda_j), 1 \leq i < j \leq k$. Each dataset retains the instances from the original dataset that belong to atleast one of the corresponding labels but not both (Table 6).

A binary classifier is then trained on each of these datasets. Also, given a new instances, it is easy to obtain a ranking of labels by first invoking all these binary classifiers and then counting their votes for each label. *Mencia et. al.* in [33] proposes *Multi-label Pairwise Perceptron (MLPP)* algorithm that uses RPC with $perceptrons$ as the binary classification method.

Table 6: Datasets transformed by RPC method

Ex.	Label
1	$\lambda_{\neg 1,2}$
2	$\lambda_{1,\neg 2}$
4	$\lambda_{\neg 1,2}$

(a)

Ex.	Label
1	$\lambda_{\neg 1,3}$
2	$\lambda_{1,\neg 3}$
4	$\lambda_{\neg 1 \neg 3}$

(b)

Ex.	Label
2	$\lambda_{1,\neg 4}$
3	$\lambda_{1,\neg 4}$
4	$\lambda_{\neg 1,4}$

(c)

Ex.	Label
4	$\lambda_{2,\neg 3}$

(d)

Ex.	Label
1	$\lambda_{2,\neg 4}$
3	$\lambda_{2,\neg 4}$

(e)

Ex.	Label
3	$\lambda_{3,\neg 4}$
4	$\lambda_{\neg 3,4}$

(f)

Calibrated Label Ranking (CLR) Even though ranking provides a relative order of the labels, *Furnkranz et. at.* in [20] argues that such ranking does not have a natural *"zero-point"* and therefore, does not provide any information about the *absolute* preference that can distinguish among all alternatives. It could be misleading to distinguish between the sets of relevant and non-relevant classes based on the label ranking. *Furnkranz et. at.* then propose *Calibrated Label Ranking (CLR)* [20] that is an extension of RPC, introducing an additional label to the original label set, which can be interpreted as a "neutral breaking point" (often called calibration label, λ_0) and can be thought as a split point between relevant and irrelevant labels. Thus a calibrated ranking,

$$\lambda_{i1} \succ \lambda_{i2} \succ .... \succ \lambda_{ij} \succ \lambda_0 \succ \lambda_{ij+1} \succ ...... \succ \lambda_{ik}$$

clearly is a ranking of the labels (ignore the calibration label λ_0) and at the same time creates a bipartition of relevant $(\lambda_{i1}..........\lambda_{ij})$ and irrelevant $(\lambda_{ij+1}......\lambda_{ik})$ labels. Each example that is annotated with a particular label, clearly is a positive example for that label and is treated as a negative example for the calibration label. Each example that is not annotated with a label is clearly a negative example for that label and is treated as a positive example for the calibration label. Thus a *Binary Relevance (BR)* classifier can then be employed to discriminate between the calibrated label and each of the other labels. Intuitively, while applied to the dataset in Table 2, CLR would work on both the data in Table 5 and Table 6, and the latter one is for the calibration label.

2.2 Simple Algorithm Adaptation Methods

Clare et. al. in [8] uses C4.5 algorithm for multi-label data with the modified entropy definition:

$$Entropy = -\sum_{j=1}^{k}\{P(\lambda_j)logP(\lambda_j) + (1 - P(\lambda_j))log(1 - P(\lambda_j))\}$$

where, $P(\lambda_j)$ = probability of class λ_j. This allows to estimate the uncertainty in terms of number of bits in multi-label setting. This modified method also allows multiple labels at the leaves.

Tree Based Boosting AdaBoost.MH and AdaBoost.MR [44] are two simple extensions of AdaBoost for multi-label data where the former tries to minimize *hamming loss* (see section 7.1.1) and the latter tries to find an hypothesis with optimal ranking. In AdaBoost.MH, examples are presented as example-label pairs and in each iterations increases the weights of misclassified example-label pairs. In contrast, AdaBoost.MR works on pairs of labels for any instance and in each iteration increases the weights of the example with mis-ordered label pairs. With a goal to produce better human readable classification rules (from trees), *Comite et. al.* [12] extends AdaBoost.MH and uses Alternating Decision Trees to train and produce rules on multi-label data.

Lazy Learning [2] There are several lazy learning (i.e. k Nearest Neighborhood (kNN)) based approaches proposed that uses either problem transformation or algorithm adaptation [66, 60, 31, 4, 47]. All these algorithms are very similar in the sense that they all uses kNN as a lazy learning approach, but what differentiates them is the aggregation of the label sets of the given instances. *Spyromitros et. al.* in [47] implements a simple method called BRkNN, that is conceptually equivalent to using *Binary Relevance (BR)* followed by kNN. However, the problem with this approach is that computational complexity becomes $|\mathcal{L}|$ times the computations cost of computing k nearest neighbors. This can easily be resolved by using single search for k nearest neighbors but at the same time making independent predictions for each label. Another important problem with BRkNN is that, if none of the labels is included in at least half of the k nearest neighbors. The traditional

[2] k in this section denotes the k parameter of the kNN algorithm, not the number of labels in $\mathcal{L}$ as elsewhere

BRkNN is tempted to output an empty label set in that case. *Spyromitros et. al.* propose two extensions to BRkNN, both based on confidence of each label, to solve this empty label set problem [47]. First, they argue that the proportion of data instances belonging to some class λ in k nearest neighbors of the test data instance is a good measure of confidence of the class label λ for that test instance.

$$Confidence(\lambda) = \frac{1}{k}\sum_{j=1}^{k} I_{N_j}(\lambda)$$

where, N is the set of labels for k nearest neighbors and,

$$I_{N_j}(\lambda) = \begin{cases} 1 & \text{if } \lambda \in N_j \\ 0 & \text{otherwise} \end{cases}$$

Therefore, in case of *empty label set* problem, they propose to return the label with highest confidence. The second extension they propose is to calculate the average size of of the label sets of the k nearest neighbors ($averagesize, s = \frac{1}{k}\sum_{j=1}^{k}|N_j|$) and anytime output s (to nearest integer value) highest confident labels for any test instance.

ML-kNN: This approach is also based on BR, but to find the label set for a given test instance it uses *maximum a posteriori* (MAP) [13], based on prior and posterior probabilities of for each k nearest neighbor label. For each instance, ML-kNN first identifies its k nearest neighbors in the training set and then poses the problem as a MAP problem below:

$$Z_t^\lambda = \arg\max_{b\in\{0,1\}} P(H_b^\lambda | E_{C_t(\lambda)}^\lambda), \qquad l \in \mathcal{L}$$

where, Z_t^λ = predicted label set for instance t, $H_1^\lambda(t)$ and $H_0^\lambda(t)$ be the events that t has label λ and does not have label λ respectively, E_i^λ is the event that exactly j instances of k nearest neighbor of test instance t has label λ and, $C_t(\lambda)$ is the membership vector that counts the number of neighbors of t belonging to class λ. This can be re-written as:

$$Z_t^\lambda = \arg\max_{b\in\{0,1\}} P(H_b^\lambda)P(E_{C_t(\lambda)}^\lambda | H_b^\lambda), \qquad l \in \mathcal{L}$$

and can be directly estimated from the training data.

Finally, *Multi-class Multi-label Associative Classification (MMAC)* [50] is an associative rule learning based covering algorithm, that recursively learns a new rule and each time removes the examples associated with the rule. Labels for the test instance are ranked according to the support of the rule that applies with the test instance. [57] extends this idea combined with lazy learning delaying the inductive process until a test instance arrives.

There also exists Neural Networks and Multi-layer perceptron based algorithms that has been extended for multi-label data. In *BP-MLL* [65], the error function for the very common neural network learning algorithm, *back-propagation* has been modified to account for multi-label data. Multi-layer perceptron is easy to extend for multi-label data where one output node is maintained for each class label. A family of online algorithms for *Multi-class Multi-layer Perceptron (MMP)* has been proposed in [11] where the perceptron algorithms weight update is performed in such a way so that it leads to correct label ranking.

Discriminative SVM Based Methods One important problem with *tree based boosting* [44] (discussed in section 2.2) is that, they are likely to overfit with relatively smaller (< 1000) training set. *Elisseeff et. al.* in [18] propose an SVM algorithm that has an intuitive way of controlling such complexity while having a small empirical error. *Godbole et. al.* present three improvements for BR with SVM to exploit label correlations that improves the margin [22]:

The first idea is to have an extended dataset with k ($= |\mathcal{L}|$) additional features which are actually the predictions of each binary classifier at the first round. The k new binary classifiers are trained on this extended dataset. In this way the extended BR takes into account potential label dependencies. The second idea, *ConfMat*, based on a confusion matrix, removes negative training examples of a complete label if it is very similar to the positive label. The third idea is called *BandSVM*, removes very similar negative examples that are within a threshold distance from the learned decision hyperplane, and this helps building better models especially in the presence of overlapping classes.

2.3 Dimensionality Reduction and Subspace Based Methods

In order to reduce the *curse of dimensionality*, features selection and extraction methods are very common in single label data. The *wrapper* based features selection [27] methods are equally applicable for multi-label data with a modified goal of reducing any multi-label loss function (see section 7). An alternative could be to transform multi-label data into single label data (e.g. BR) and then apply features selection to evaluate the discriminative power of each feature with respect to each label. Any unsupervised method for dimensionality reduction such as, *Principal Components Analysis (PCA)* and *Latent Semantic Indexing (LSI)* can also be used with multi-label data without any modification. However, *Yu et. al* argue that even though such unsupervised algorithms can be used directly, if the relevant class label information is available, then taking that into account while deriving the objective function should be beneficial [64]. Based on this idea, they propose a *Multi-label Informed Latent Semantic Indexing (MLSI)* that preserves the feature information as well as captures the label correlations, while posing the LSI problem as an optimization problem.

On the other hand, supervised subspace based methods require modifications to be applicable to multi-label data. A multi-label version of *Linear Discriminant Analysis (LDA)* has been proposed in [38] where the objective function, the ratio of inter-class distance to intra-class distance has been adapted for multi-label data.

Shared Subspace *Shared subspace* based approaches assumes that there exist a common subspace that is shared among the class labels. Therefore, any algorithm that intend to extract the shared subspace and the decision function is learnt on that subspace not only reduce the information redundancy but also is able to take into account the label correlations. *Yan et. al.* propose a boosting algorithm called *Model-shared Subspace Boosting (MSSBoost)* [62]. This method uses *random subspace sampling* to select feature subspace to work on and then, model sharing techniques to automatically find, share and combine a number of random subspaces, and finally boosting based ensemble learning to jointly optimize the loss function over all labels. A much simpler method for the same purpose is proposed in [25] that uses a linear transformation method to capture the shared subspace and finally uses a binary classifier train on this shared subspace.

2.4 Ensemble Methods[3]

The *Random k labelsets (RAkEL)* proposed by *Tsoumakas et. al.* [54] is an ensemble method that iteratively constructs an ensemble of some (# of models desired, m) *Label Powerset(LP)* clasifiers. At each iteration it randomly selects (without replacement) k-labelset and learns and LP classifier and finally outputs all classifiers learnt. Notice, m and k are two important parameters, and the claim is that for small values of k and adequately large m, RAkEL is able to model label correlations effectively. Notice, $k = 1$ and $m = |\mathcal{L}|$ would imply each time training an LP classifier and therefore a total number for $\mathcal{L}$ classifiers are trained, one for each class; this is *Binary Relevance (BR)*. Similarly, $k = \mathcal{L}$ and $m = 1$ would turn it into a single global LP classifier.

To address the class imbalance problem (very few instances in some classes) in LP *Read et. al.* in [42] proposes *Pruned Sets* method. The idea is based on pruning away infrequently occurring label sets in LP, and this allows to focus only on most important label correlations, with reduced complexity. To compensate for such information loss, pruned away sets are broken into more frequently occurring subsets and is introduced in the data again once crosses some predefined threshold. An ensemble of such *pruned set* classifiers is also proposed and for classification of a new instance a *voting scheme* is followed.

Zhang et. al. [69] present a *Random Decision Tree (RDT)* based ensemble and demonstrate that the training complexity is independent from the number of class labels. RDT constructs several decision tree randomly (i.e. picks a remaining feature randomly at each node) and stops growing once it crosses some predefined threshold. Theoretical risk analysis of RDT shows that the upper bound of the risk is stable and lower bound decreases with the increase of number of trees [69]. Importantly, increased number of trees in RDT guarantees improved performance with slightly increased complexity, though such increased complexity is better than most other methods. This idea of RDT can be used with both LP and BR and in both cases the computational complexity is

[3] k in this section is the parameter for RAkEL

free of number of labels. This property makes it better than the hierarchical model based algorithm HOMER [53] (see section 4), in terms of both performance and complexity.

2.5 Generative Modeling for Multi-label Data[4]

Several generative modeling based approach had been proposed [32, 56, 59] especially for modeling the generative process of text based multi-label data. $McCallum$ in [32] proposes a generative model for multi-labelled document that starts with selecting a set of classes (class labels for a document) with a mixture of weights for those classes. Next, each word in the document is generated by first selecting a class according to the weight distribution and then the selected class is allowed to generate a word. Using the single apriori on most probable weight vector, $\vec{\Psi}^{\vec{c}}$, given the set of classes $\vec{c}$, an approximate closed form generative process has been defined as:

$$P(\vec{c}|d) \approx P(\vec{c}) \prod_{w \in d} \sum_{c \in C} \vec{\Psi}^{\vec{c}}_{c} P(w|c)$$

Finally, an Expectation Maximization(EM) [16] based parameter estimation method has been proposed for the above model. A similar word based *parametric mixture models (PMM)* has been presented in [56] that directly models the probability of each word appearing in each document belonging to some class.

To benefit from the correlations among classes *Wang et. al.* presented generative *Correlated Labeling Model (CoL Model)* that captures the underlying structures via the latent random variables in a supervised fashion [59]. Figure 1 shows the graphical representation of the CoL model and the joint distribution is given by:

$$P(\varphi, \theta, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{c}, \boldsymbol{w}) = P(\varphi|\mu, \Sigma) \prod_{i=1}^{k} P(c_i|\varphi) P(\theta_i|\alpha, c_i) \prod_{j=1}^{n} P(y_j|k) P(z_j|\theta_{y_j}) P(w_j|\beta, z_j)$$

where, φ = document specific distribution of classes given by a multivariate normal distribution, θ = distribution of topic under each class, $\boldsymbol{y}$ = indexing of words to indicate which class generates the corresponding topic, $\boldsymbol{z}$ = topic assignment for each word, $\boldsymbol{c}$ = $\{0, 1\}^k$ class membership vector, k = total number of classes, $\boldsymbol{w}$ = observed words in a document, n = total number of words in a document, α = Dirichlet parameters for topic prior distribution under each class, β = multinomial parameters to represent the word distribution under topics. While this model is very similar to *Latent Dirichlet Allocation (LDA)* [2] based topic model, differences include: CoL model is supervised and LDA model is unsupervised, CoL model uses class membership which LDA model does not and so on. A variational inference method is also presented in [59] to estimate the parameters of this CoL model.

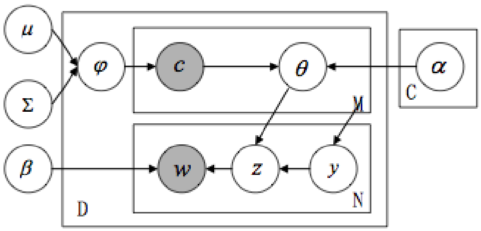

Figure 1: Graphical representation of CoL Model (Figure taken from [59])

A conditional random field (CRF) based model is presented in [21] where two multi-label graphical models has been proposed, both parameterizes label co-occurances. The *Collective Multi-label (CML)* classifier maintains feature accounting for label co-occurances and the *Collective Multi-label with Features (CMLF)* maintains parameters that correspond to features for each co-occuring label pair. *Petterson et. al.* recently presented another interesting generative modeling approach in a *reverse manner*, predicting a set of instances given the labels [39].

3 Exploiting Label Structure

While some of the learning methods above does incorporate label correlations into account during training, a number of methods (e.g. BR,) are criticized for assuming label independence. One simple method that takes into account this label dependence is the *Hierarchical Binary Relevance*

[4] this section uses symbols and notations that are commonly used in topic model

(HBR) (given a hierarchy, traing a binary classifier for each non-root class) method, where an implicit constraint is that an instance cannot belong to a class if it is not associated with its parent. Similar hierarchical approach has been utilized in all tree based methods (e.g. Tree Based Boosting, HOMER etc.). All these methods are capable of utilizing label dependencies reflected in their improved predictive accuracies.

However, it is important to note that there exists two different kinds of label dependence: *conditional* and *unconditional*, and substantially different approaches are needed to exploit these different dependencies [15]. *Zhu et. al.* in [71] present a *maximum entropy* based method that explicitly models the mutual correlations among the classes by constructing a conditional probability model from the training data. Unlike other traditional approaches, the conditional probability model parameters here are estimated through *maximum entropy* method subject to the prior probabilities of each category (i. e. $E_e(\lambda) = E_m(\lambda) + \eta, E = expectation, e = empirical, m =$*model distribution*, $\eta = error$) and correlations among the categories and the features of the given data (i. e. $E_e(\lambda|x_j) = E_m(\lambda|x_j) + \phi$), and $E_e(\lambda_i\lambda_j) = E_m(\lambda_i\lambda_j) + \theta_{ij}, \phi = error, \theta_{ij} = error$) and finally, solved using *Lagrangian*. A similar approach is presented in [67] that uses *Bayesian Network* to encode the label dependencies.

To estimate the joint distribution of labels a completely different idea proposed in [14] is to learn k different function ($k = |\mathcal{L}|$) on extended input space $\mathcal{X} \times \{0,1\}^{i-1}$, where predictions for classes, $y_1, y_2, ..., y_{i-1}$ (notice that this idea is very similar the SVM based approach presented in 2.2) are considered as additional features.

$$f_i : \mathcal{X} \times \{0,1\}^{i-1} \to [0,1]$$
$$(\mathbf{x}, y_1, y_2, ..., y_{i-1}) \to P(y_i = 1|\mathbf{x}, y_1, y_2, ..., y_{i-1})$$

where, f_i can be interpreted as a *probabilistic classifier* and this approach is called *probabilistic classifier chain (PCC)*. The idea is further extended to construct an ensemble, called *probabilistic classifier chain (EPCC).*

Encoding in Input Space A completely different approach called the *Instance Differentiation (INSDIF)* [68] is based on the idea that multi-label learning can be better efficient if the input space inherent ambiguity can be expressed explicitly. The two stage algorithm *first* computes a prototype vector for each class by averaging all instances belonging to that class and then each instance is represented by a bag of instances, that are the differences between the instance itself and each of the prototype vectors. The *second* step is to use a two level classification strategy: using k-medoids algorithm to find a fixed number of clusters representing the underlying structure of the data and then finding a discriminative linear classifier weights based on the features computed by using the distance between the bag of instances and the cluster medoids (notice that the second step is similar to solving *Multi-Instance Multi-label* problem discussed in section 6.3).

4 Scaling with Large Data

Often a number of domains and applications require multi-label algorithms to scale up with large data such as $delicious$, EUROVOC (see section 8.2), and the problem space could be nearly *unbounded* if we consider the categorization of the web. Such large domains pose a few challenges to the existing algorithms, especially because the label space is also likely to grow with the exponential growth of instance space. *First*, with a large label space, number of training examples labeled for a particular class will be significantly less compared to the total number of examples. *Second*, the computational cost of training a multi-label classifier is usually strongly affected by the number of labels. This is true for most of the existing algorithms except for a few such as *binary relevance* whose complexity is linear with respect to $|\mathcal{L}|$, but usually criticized for label independence assumption. *Finally*, most of the methods need to maintain a large number of models in memory and, therefore, may fail to scale with large label space.

Most of the existing algorithms described above either do not scale or result in unsatisfactory performance. *Tang et. al.* in [49] present a *MetaLabeler* approach that first constructs a $meta$ dataset where each data instance is a function of corresponding instance in the original dataset. Three different such function has been proposed: *content-based* ($\phi(x) = x$), *score-based* ($\phi(x) = [f_1(x), ..., f_k(x)]$, $f_i(x)$ = prediction score for class λ_i) and, *rank-based* ($\phi(x) =$

$\psi(f_1(x), ..., f_k(x)), \psi(x) =$ a vector of sorted scores). One-vs-rest SVM model is trained on this *meta* dataset and finally, the class labels are selected based on their scores.

Dekel et. al. propose a *pruning* based two step method, which is very similar to the *ensemble pruned set* method discussed above (see section 2.4). Theoretically and experimentally they have shown that this method is capable of dealing very sparse situation even with more classes than instances [36]. In the *learning* step, any arbitrary *classifier, h* is trained on the training data. In the *post learning* step, the goal is to find a *label transformation* function, ϕ so that the final classifier is the composite of the two, $\phi \circ h$. This label transformation in this case is defined as a set of *label pruning rules* that minimizes the overall risk of $\phi \circ h$. A theoretical analysis of *final empirical risk*, $\hat{R}_\phi$ and the *final risk*, R_ϕ shows that a simple pruning rule as, *'pruning any label for which the ratio of false positive to true positive exceeds some threshold'*, reduces the overall risk under mild conditions. More importantly, unlike most other methods, this method allow *number of labels, k* to grow as a function of *number of instances, n*. While this method seem to work competitively good even with large domain, *Wikipedia*, with 2.9 million articles and almost 1.5 million categories, a potential criticism against this method is the implicit *label independence* assumption. Also, incorporating more complicated pruning rules in the system is left as a future research question.

Hierarchical Methods The idea behind hierarchical model comes from the goal to reduce the computational complexity at each level and thus making the learning algorithm efficient especially with many labels. While the *tree based boosting* (see section 2.2) and *random decision tree* (see section 2.4) based methods are very similar to the hierarchical method HOMER proposed in [53], what distinguishes HOMER from other methods is the *balanced clustering* [1] to explicitly maintain an even distribution of a set of labels into disjoint subsets so that similar labels are placed together and dissimilar apart. HOMER starts with all the labels (and instances) at the root and recursively creates an hierarchy in a top-down depth first fashion. At each node, b children nodes (b = user defined branching factor) are created and the labels of the current node are distributed into b disjoint subsets, one for each child. Such distribution of labels among the children is performed using a *balanced clustering* method. They propose a *Balanced k means* algorithm which is an extension of popular *k means* algorithm with an explicit constraint on the size of each cluster. Using such balanced label clusters would ensure reduced operational cost in terms of number of nodes to be activated for any prediction. Once the balanced distribution of labels is done, a multi-label classifier is trained for the prediction of the meta-labels (disjunction of the labels at any node) of the children at current node.

5 Online Multi-label Learning

Many real world multi-label applications such as web text categorization, content based image or video annotation etc., require not only dealing with large volume of data but also demand for processing or update of the learning algorithm in an online fashion [61]. Unfortunately, most of the algorithms described above does not scale well with large data and more importantly are not really applicable for online learning. To address this issue, *Hua et. al.* propose a scalable framework for annotation based video search that relies on on an *online multi-label active learning (Online MLAL)* [23]. *Online MLAL* has three major modules: *multi-label active learning*, *online multi-label learning*, and *new label learning from zero knowledge*. The *first* one is a simple active learning module that selects the *most informative instance-label pair* (x_s^*, λ_s^*) to reduce the uncertainty along both instance and label dimensionalities, based on *Bayesian* classification error over a sample of all instances. The assumption for *Online MLAL* is that the data is increasing batch by batch and once and an initial classifier is build on the first batch, it can be updated through active learning over succeeding batches. The goal for the *second* module is to make this algorithm run online with two specific requirements: the classifier state does not change much from its current state, and reveals the information contained in the newly arrived instances. The first one can be measured in terms of *Kullback Leibler Divergence (KLD)* [9] while ensuring the second one in terms of reducing bayesian error; thus can be posed as an optimization problem subject to constraints. *Finally*, such classification scheme is extended to handle new labels (can be proposed from the query logs, if accessible) assuming the new labels are uniformly distributed.

In [61] *Zhang et. al.* present *Bayesian Online Multi-label (BOMC)* classification framework that uses a probabilistic linear discriminant w_c for each class c where $\{w_c\}_{c \in C}$ are independent diagonal

Gaussians, whose mean and variance are estimated from the training data. The key model here is the likelihood $P(\mathbf{y}|\{w_c\}_c, \mathbf{x})$ that is modeled using a factor graph. Also the posterior $P(\{w_c\}|\mathbf{y}, \mathbf{x})$ can be estimated by marginalizing over different nodes (inner product nodes, noise additive nodes and the difference nodes) in the factor graph (for details see [61]). At each iteration, it takes one additional instance (x_i, y_i) and using a *Gaussian* prior $P_0(\mathbf{w})$, computes the posterior:

$$P_i(\mathbf{w}|\mathbf{x}_i, \mathbf{y}_i) \propto P_{i-1}(\mathbf{w})P(\mathbf{y}_i|\mathbf{w}, \mathbf{x}_i)$$

This can actually be approximated by a *Gaussian* that is closest in terms of *KL-divergence (KLD)*. Finally, such marginalization in subsequent nodes can be effectively performed by *Expectation Propagation (EP)* [35] through the factor graph.

6 Problem Variations

It is worth mentioning that the most popular *single label multi-class classification* (often the term *single label* is ignored) is a simple variation of multi-label classification problem, where the constraint is that every instance can belong to only one class. That reduces the problem to learn a classifier, $h_{multi-class} : \mathcal{X} \rightarrow \mathcal{L}$. In this section, we describe a few more interesting problems that closely relates to multi-label problem.

6.1 Learning with Multiple Labels: Disjoint Case

Consider an example scenario where several human experts are hired to label some data into different categories. While we expect that they would agree in most (if not all) of the labels, they might disagree is some of the instances' labels (e.g. consider a complicated labeling such as labeling emotions from image/video). This might result in multiple class labels for some instances while only one of them is actually correct. This problem is similar to traditional *multi-label* problem because it also allows multiple labels for each instance, however, quite different because only one of those labels (for an instance) is correct while in case of multi-label, all the labels are correct [26].

Definition 6.1 *Learning with Multiple Labels (Disjoint Case): Let $\boldsymbol{x}_i$ be an instance, $\boldsymbol{Y}_i$ be the set of candidate class labels for $\boldsymbol{x}_i$. Assuming that model with parameter space Θ exists (that maps inputs to correct output labels), the goal for learning from multiple label data (disjoint case) is to estimate $\theta \in \Theta$ so that the predicted class z_i^l for data instance x_i is highly likely to be in $\boldsymbol{Y}_i$.*

$$\theta^* = \arg\max_{\theta} \prod_i P(z_i^l \in \boldsymbol{Y}^l | \boldsymbol{x}_i, \theta)$$

Notice that this problem is very similar to *semi-supervised* learning. Like *semi-supervised* learning this problem has a portion of the data correctly labeled (labeled with only one class). But the rest of the data in case of *semi-supervised* learning is unlabeled and, therefore, the label search space is $\mathcal{L}$ (set of all labels) for those instances; whereas, in this case the search space is restricted to the given labels for those instances.

Finally, *Jin et. al.* in [26] proposes a discriminative Expectation Maximization (EM) [16] based algorithm to estimate θ^*, with and without prior assumption.

6.2 Multitask Learning

In *machine learning*, it is often common to break down a large problem in to a set of small and reasonably independent subproblems, learn them separately and them combine [5]. A counter argument for this approach is that this method ignores a lot of information and possible dependence among the subproblems. For example, if we train a system to learn textures in isolation, that might fail in real scenario when tested with complex real-life scenes. In contrast, a system trained simultaneously on textures, shades, shapes, reflections, shadows, orientations, edges, etc. can potentially benefit from inter-dependence among all these and is expected to perform better to recognize complex objects in the real world. Intuitively, this approach is called *multitask learning* where the basic idea is that, if several tasks are related, then learning them *simultaneously* can improve the overall performance compared to learning each of them separately [40].

Definition 6.2 *Multitask Learning (MLT): Let $\mathbf{X}_i = (x_{i1}, x_{i2}....., x_{im_i}) \subseteq \mathcal{X}_i$ be a set of instances in the instance space $\mathcal{X}_i$, $\mathbf{Y}_i^l = (y_{i1}^l, y_{i2}^l....., y_{ip_i}^l)$, $y_{ij}^l \subseteq \mathcal{L}_i$, is a set of labelsets for each instance space i, $1 \leq i \leq v$, v = total number of spaces (different tasks), m_i = total number of instances in space i, p_i = total number of different labels in space i. Now, if learning a task is actually to learn a function from the input space $\mathcal{X}$ to the output space $\mathbf{Y}^l$, then multitask learning is the problem of learning several functions $f_i : \mathcal{X}_i \rightarrow \mathbf{Y}_i^l$.*

The assumption here is, that the observations are disjoint but they are drawn from the same domain, i.e. $\mathcal{X}_1 = \mathcal{X}_2 = ... = \mathcal{X}_v = \mathcal{X}$. It is also common to assume that the set of labels $\mathbf{Y}_i^l$ are same across all tasks, or we have access to some oracle that maps among labels in different spaces. One practical example[5], is the problem of learning to automatically categorize objects on the web in to an ontology or directory structure. Two most common applications for this problem are *Yahoo!* directory[6] and DMOZ [7]. Clearly, these two tasks are related, but they might differ in their label sets. Having access to an oracle for mapping among label sets might not be a practical assumption here, still, learning the two problems simultaneously can greatly be benefitted by learning them simultaneously.

There exists several methods in literature to address this problem of learning multiple tasks simultaneously [30, 10]. $Mencia$ in [30] argues that it would advantageous to consider all the parallel subproblems as a single large problem and poses it as a *multi-label* problem, assuming that there is a common training set for all tasks.

6.3 Multi-Instance Multi-Label Learning (MIML)

In *multi-instance multi-label* problem, instances are organized into *'bags'* that may contain multiple instances and the class label is assigned for each bag instead of each instance. The only constrain for labeling each bag is that it needs to have at least one instance belonging to that class and the rest could be noise for that class. It is important to note that for multi-label problems, the ambiguity lies on the class label, but for multi-instance the ambiguity lies on the instances side [26]. This makes *multi-instance multi-label* problems more difficult as they can have ambiguity in both sides.

Definition 6.3 *Multi-instance Multi-label Learning (MIML): Let $\mathcal{X}$ be the instance space and $\mathcal{L}$ be the set of labels. Given a dataset $\{(X_1, Y_1^l), (X_2, Y_2^l), ..., (X_n, Y_n^l)\}$ where $X_i \subseteq \mathcal{X}$ is a set of instances $\{x_{i1}, x_{i2}, ..., x_{in_i}\}$, $x_{ij} \in \mathcal{X}$, $1 \leq i \leq n, 1 \leq j \leq n_i$, and $Y_i^l \subseteq \mathcal{L}$ is a set of labels $\{y_{i1}^l, y_{i2}^l, ..., y_{il_i}^l\}$, $y_{ik}^l \in \mathcal{L}$, $1 \leq k \leq l_i$, where n_i denotes the number of instances in X_i and l_i denotes the number of labels in Y_i^l.*

*If each Y_i^l contains only one label ($Y_i^l = \{y_i^l\}$), then data is called **multi-instance data** and learning a function $f_{MIL} : 2^{\mathcal{X}} \rightarrow \mathcal{L}$ $\big($in other words $2^{\mathcal{X}} \times \mathcal{L} \rightarrow \{+1, -1\}\big)$ is called **multi-instance learning**.*

*Learning a function $f_{MIML} : 2^{\mathcal{X}} \rightarrow 2^{\mathcal{L}}$ is called **multi-instance multi-label learning**[8].*

There are two common approaches to solve MIML problems [70]:

Using Multi-instance Learning: For any $y \in \mathcal{L}$, $f_{MIL}(\mathbf{X}_i, y) = +1$ if $y \in Y_i^l$ and -1 otherwise. Now the appropriate label for a new instances X* can be computed as, Y* = $\{y | y \in \mathcal{L} \texttt{ and } f_{MIL}((X*, y)) = +1\}$. Thus, we can transform $f_{MIML} : 2^{\mathcal{X}} \rightarrow 2^{\mathcal{L}}$ into $f_{MIL} : 2^{\mathcal{X}} \times \mathcal{L} \rightarrow \{+1, -1\}$.

Using Multi-label Learning: Let ϕ is a function such that $\pi_i = \phi(X_i)$, $\phi : 2^{\mathcal{X}} \rightarrow \Pi$. Now, for $\pi_i \in \Pi$ we can learn a multi-label algorithms such that, $f_{MLL}(\pi_i) = f_{MIML}(X_i)$, where, $f_{MLL} : \Pi \rightarrow 2^{|\mathcal{L}|}$. Thus, we can transform $f_{MIML} : 2^{\mathcal{X}} \rightarrow 2^{\mathcal{L}}$ into $f_{MLL} : \Pi \rightarrow 2^{|\mathcal{L}|}$.

[5] example borrowed from [40]

[6] http://dir.yahoo.com/

[7] http://www.dmoz.org/

[8] notice we have already defined multi-label learning in definition 2.1

7 Evaluation Metrices

In traditional classification such as multi-class problems, *accuracy* is the most common evaluation criteria. Additionally, there exists a set of standard evaluation metrics thats includes *precision, recall, F-measure*, and *ROC area* defined for single label multi-class classification problems[19]. However, in multi-label classification, predictions for an instance is a set of labels and, therefore, the prediction can be *fully correct, partially correct* (with different levels of correctness) or *fully incorrect*. None of these existing evaluation metrics capture such notion in their original form. This makes evaluation of a multi-label classifier more challenging than evaluation of a single label classifier.

Depending on the target problem, evaluation measures for multi-label data can be grouped in to atleast three groups: *evaluating partitions, evaluating ranking and using label hierarchy* [55]. The *first* one evaluates the quality of the classification in to classes, the *second* one evaluates if the classes are ranked in order relevance and the *third* one evaluates how effectively the learning system is able to take into account an existing hierarchical structure of the labels.

Let T be a multi-label dataset consisting n multi-label examples $(\mathbf{x}_i, \mathbf{Y}_i)$, $1 \leq i \leq n$, $(\mathbf{x}_i \in \mathcal{X}, \mathbf{Y}_i \in \mathcal{Y} = \{0,1\}^k)$, with a labelset $\mathcal{L}, |\mathcal{L}| = k$. Let h be a multi-label classifier and $Z_i = h(\mathbf{x}_i) = \{0,1\}^k$ be the set of label memberships predicted by h for the example $\mathbf{x}_i$.

7.1 Partitions

Evaluation of some learning algorithm is a measurement of how far the learning system predictions are from the actual class labels, tested on some unseen data. To capture the notion of *partially correct*, one strategy is to evaluate the *average difference* between the predicted labels and the actual labels for each test example, and then average over all examples in the test set. This approach is called *example based* evaluations. Seemingly, one could define a *label based* evaluation where each label is evaluated first and then averaged over all labels. It is important to note that any such *label based* method would fail to directly address the correlations among different classes [67].

7.1.1 Example-based

Exact Match Ratio (MR): As described above, evaluation of a multi-label classification algorithm is difficult mostly because mutli-label prediction has an additional notion of being *partially correct*. One trivial way around would be just to ignore *partially correct* (consider them as *incorrect*) and extend the *accuracy* used in single label case for multi-label prediction. This is called *Exact Match Ratio*.

$$ExactMatchRatio, MR = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = Z_i)$$

where, *I* is the indicator function. Clearly, a disadvantage of this measure is that it doesn't distinguish between *complete incorrect* and *partially correct* which might be considered as *harsh*.

In order to account for *partially correctness*, *Godbole et. at* in [22] proposed following set of definitions for *accuracy, precision, recall,* and F_1 *measure*.

Accuracy (A): *Accuracy* for each instance is defined as the proportion of the predicted *correct* labels to the total number (predicted and actual) of labels for that instance. Overall $accuracy$ is the average across all instances.

$$Accuracy, A = \frac{1}{n}\sum_{i=1}^{n} \frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}$$

Precision (P): *Precision* is the proportion of predicted correct labels to the total number of actual labels, averaged over all instances.

$$Precision, P = \frac{1}{n}\sum_{i=1}^{n} \frac{|Y_i \cap Z_i|}{|Z_i|}$$

Recall (R): *Recall* is the proportion of predicted correct labels to the total number of predicted labels, averaged over all instances.

$$Recall, R = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i|}$$

F_1-Measure (F): Definition for $precision$ and $recall$ naturally leads to the following definition for F_1-*measure* (harmonic mean of precision and recall [19]).

$$F_1 = \frac{1}{n}\sum_{i=1}^{n}\frac{2|Y_i \cap Z_i|}{|Y_i| + |Z_i|}$$

As in single label multi-class classification, the higher the value of *accuracy, precision, recall* and F_1- score, the better the performance of the learning algorithm.

Hamming Loss (HL): *Hamming Loss* reports how many times on average, the relevance of an example to a class label is incorrectly predicted [44]. Therefore, *hamming loss* takes into account the prediction error (an incorrect label is predicted) and the missing error (a relevant label not predicted), normalized over total number of classes and total number of examples.

$$HammingLoss, HL = \frac{1}{kn}\sum_{i=1}^{n}\sum_{l=1}^{k}[I(l \in Z_i \wedge l \notin Y_i) + I(l \notin Z_i \wedge l \in Y_i)]$$

where I is the indicator function. Ideally, we would expect *hamming loss, HL = 0*, which would imply no error; practically the smaller the value of *hamming loss*, the better the performance of the learning algorithm.

If $|Y_i| = 1$ and $|Z_i| = 1$, then we have a single label multi-class classification problem. It is easy to note that, in that case, *hamming loss* is $\frac{2}{k}$ times of the classification error [66].

7.1.2 Label-based

Label based measures evaluate each label separately and then averages over all labels. Therefore, any known measure, used for evaluation of a binary classifier (e.g. accuracy, precision, recall, F_1, ROC etc.), can be used here. Any of these scores can be computed on individual class labels first and then averaged over all classes. In contrast, they can be computed globally over all instances and all class labels. The first one is called *macro averaging* and the second one is called *micro averaging* [63]. Below are the definitions for macro and micro averaged precision, recall and F_1.

Macro Averaged Measures:

$$\lambda - Precision, P^{\lambda}_{macro} = \frac{\sum_{i=1}^{n} Y_i^{\lambda} Z_i^{\lambda}}{\sum_{i=1}^{n} Z_i^{\lambda}} \qquad Precision, P_{macro} = \frac{1}{k}\sum_{i=1}^{k} P^{\lambda}_{macro}$$

$$\lambda - Recall, R^{\lambda}_{macro} = \frac{\sum_{i=1}^{n} Y_i^{\lambda} Z_i^{\lambda}}{\sum_{i=1}^{n} Y_i^{\lambda}} \qquad Recall, R_{macro} = \frac{1}{k}\sum_{i=1}^{k} R^{\lambda}_{macro}$$

$$\lambda - F^{\lambda}_{1-macro} = \frac{2\sum_{i=1}^{n} Y_i^{\lambda} Z_i^{\lambda}}{\sum_{i=1}^{n} Y_i^{\lambda} + \sum_{i=1}^{n} Z_i^{\lambda}} \qquad F_{1-macro} = \frac{1}{k}\sum_{i=1}^{k} F^{\lambda}_{macro}$$

Micro Averaged Measures:

$$Precision, P_{micro} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n} Y_i^{j} Z_i^{j}}{\sum_{j=1}^{k}\sum_{i=1}^{n} Z_i^{j}}$$

$$Recall, R_{micro} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n} Y_i^{j} Z_i^{j}}{\sum_{j=1}^{k}\sum_{i=1}^{n} Y_i^{j}}$$

$$F_{1-micro} = \frac{2\sum_{j=1}^{k}\sum_{i=1}^{n} Y_i^j Z_i^j}{\sum_{j=1}^{k}\sum_{i=1}^{n} Y_i^j + \sum_{j=1}^{k}\sum_{i=1}^{n} Z_i^j}$$

where

$$Y_i^{\lambda} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ actually belongs to class } \lambda \\ 0 & \text{otherwise} \end{cases}$$

and,

$$Z_i^{\lambda} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ is predicted to be belong to to class } \lambda \\ 0 & \text{otherwise} \end{cases}$$

It is important to note here that by definition, macro averaged F_1 would be more affected by the performance of the classes which has fewer examples and in contrast, micro averaged F_1 would be more affected by the performance of the classes which has more examples [49].

α ***-Evaluation:*** *Boutell et. al.* in [3] proposes a more sophisticated and generalized framework for all of the above evaluation metrics, with a control of forgiveness to classification errors and missing errors separately. Let, classification error, $C_i = Z_i \bigoplus Y_i$ and missing error, $M_i = Y_i \bigoplus Z_i$ where,

$$A \bigoplus B = \{A[j]^{A[j]-B[j]}\}, 1 \leq j \leq k$$

Then each prediction in α-evaluation can be scored by the following formula:

$$score(Z_i) = \left(1 - \frac{|\beta M_i + \gamma C_i|}{|Y_i \ \cup Z_i|}\right)^{\alpha}$$

where, $\alpha \geq 0, \beta \geq 0, \gamma \leq 1, \beta = 1 | \gamma = 1$. These parameters allow classification error and missing error to be penalized differently, *adjusted to specific problem type*. If we penalize both error type equally like in all cases above, then the equation reduces to:

$$score(Z_i) = \left(\frac{|M_i \cap C_i|}{|Y_i \ \cup Z_i|}\right)^{\alpha}$$

where $\alpha \geq 0$. Here, α controls the rate of forgiveness for any mistake made on any prediction. A smaller value for α would result in an aggressive (tend to forgive error, meaning, it would allow some score for partially correct labels) approach and a large value for α would result in conservative (tend to allow very low score to partially correct labels) approach. Now we can define the evaluation metrics using this score:

$$Accuracy, A_{score} = \frac{1}{n}\sum_{i=1}^{n} score(Z_i)$$

$$Precision(\lambda), P_{score}^{\lambda} = \frac{1}{D_{\lambda}^{Y}} \sum_{x \in D_{\lambda}^{Y}} score(Z_x)$$

$$Recall(\lambda), R_{score}^{\lambda} = \frac{1}{D_{\lambda}^{Z}} \sum_{x \in D_{\lambda}^{Z}} score(Z_x)$$

where, D_{λ}^{Y} and D_{λ}^{Z} denotes all the instances for which λ is the actual class and predicted class respectively.

7.2 Rankings

If a classifier is able to learn the ranking of the predicted labels, then the following metrics are common to evaluate the performance of the algorithm [55].

One Error (O): *One error* measures how many times the top ranked predicted label is not in the set of true labels of the instance.

$$One - error, O = \frac{1}{n}\sum_{i=1}^{n} I(\arg\min_{\lambda \in \mathcal{L}} r_i(\lambda) \notin Y_i^l)$$

where, I is an indicator function and $r_i(\lambda)$ is the predicted rank of class label λ for an instance x_i. The top ranked predicted label is the label the classifier is most confident on and getting it wrong would clearly be an indication of overall lower performance of the classifier. Ideally, we would expect the perfect performance when *one error = 0*; practically the smaller the value of *one error*, the better the performance. *One error* is the *classification error* for single label classification.

Coverage (C): For some applications, it is often important to get all the true labels predicted even with a few extra *false positive* predicted labels (e.g. fraud detection). $Coverage$ is the metric that evaluates how far on average a learning algorithm need to go down in the ordered list of prediction to cover all the true labels of an instance. Clearly, the smaller the value of $coverage$, the better the performance.

$$Coverage, C = \frac{1}{n}\sum_{i=1}^{n}\max_{\lambda\in Y_i} r_i(\lambda) - 1$$

Ranking Loss (RL): Instead of comparing two label subsets, *ranking loss* evaluates the average proportion of label pairs that are incorrectly ordered for an instance.

$$RankingLoss, RL = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Y_i^l||\overline{Y_i^l}|}|(\lambda_a,\lambda_b) : r_i(\lambda_a) > r_i(\lambda_b), (\lambda_a,\lambda_b) \in Y_i^l \times \overline{Y_i^l}|$$

where, $\overline{Y_i} = \mathcal{L} \setminus Y_i$. Similar to *one error*, the smaller the *ranking loss*, the better the performance of the learning algorithm.

Average Precision (AP): For each relevant label, average precision computes the proportion of relevant labels that are ranked before it, and finally averages over all relevant labels.

$$AveragePrecision, AP = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Y_i^l|}\sum_{\lambda\in Y_i^l}\frac{|\{\lambda^{'} \in Y_i^l : r_i(\lambda^{'} \leq r_i(\lambda))\}|}{r_i(\lambda)}$$

This idea is borrowed from *Information Retrieval* where *average precision* is used to evaluate document ranking performance for a given query [43]. The higher the value of *average precision*, the better the performance and *average precision* = 1 means the perfect performance.

7.3 Hierarchical

HIerarchical loss extends *hamming loss* to account for any existing underlying hierarchical strucuture of the labels [7]. The idea of *hierarchical loss* is based on the notion that, whenever a classifier makes a mistake at any node in a given hierarchy, no further loss should be counted for any mistake in the subtree rooted at that particular node. An easy way to do this would be to parse the predicted label hierarchy in a top down (e.g. pre-order) manner and ignore any subtree which is rooted at a wrong prediction node.

$$H - Loss = \frac{1}{m}\sum_{i=1}^{n}|\lambda|\lambda \in (Y_i \bowtie Z_i) \wedge Ancestor(\lambda) \cap (Y_i \bowtie Z_i) = \phi|$$

where, $\bowtie$ denotes the symmetric difference between two sets.

8 Multi-label Datasets and Statistics

Before running any experiment with any multi-label data, it is important to notice that not all multi-label datasets are equal even with same number of instances or with same number of labels [55]. Number of labels for each example can be variable across different datasets and this could influence the performance of different multi-label learning algorithms. Therefore, it would unfair to compare different methods without noting the dataset properties.

8.1 Dataset Properties

Given a multi-label dataset, $S = (\mathbf{x_i}, Y_i)$, $1 \leq i \leq n$, following data properties can be defined to compare different sets of data.

Table 7: Scene Data Distribution [3]

Label Set	#Images	Label Set	#Images	Label Set	#Images
Beach	369	Urban	405	Fall Foliage + Mountain	13
Sunset	364	Beach + Field	1	Field + Mountain	75
Fall Foliage	360	Beach + Mountain	38	Field + Urban	6
Field	327	Beach + Urban	19	Mountain + Urban	1
Mountain	405	Fall Foliage + Field	23	Fall Faliage + Field + Mountain	1

Definition 8.1 *Distinct Label Set (DL) is the total count of number of distinct label combinations observed in the dataset [67].*

$$DL = |\{Y|\}|\exists \mathbf{x} : (\mathbf{x}, Y) \in S|$$

Notice that a high DL for any dataset would simply imply a large label space and an upperbound for this label space is $2^{|L|} - 1$.

Definition 8.2 *Proportion of Distinct Label Set (PDL) is the normalized Distinct Label Set, normalized by total number of examples [67].*

$$PDL = \frac{DL}{|S|}$$

Therefore, PDL is the measure of number of distinct labelsets per example.

Definition 8.3 *Label Cardinality (LCard) is the average number of labels per example [55].*

$$LCard = \frac{1}{n}\sum_{i=1}^{n}|Y_i|$$

An upperbound for LCard is $|L|$. $Read\ et.\ al.$ *in [42] calls LCard as a measure of "multi-labelledness".*

Definition 8.4 *Label Density (LDen) is LCard normalized by the the number of labels [55].*

$$LDen = \frac{LCard}{k}$$

Such inherent dataset properties can cause different algorithms to perform differently on different datasets, based on the underlying assumptions of the algorithms. The following section discusses about different benchmark datasets and Table 8 shows differences among different datasets in terms of above mentioned properties.

8.2 Benchmark Datasets

Emotions Dataset It has been observed that music signals belonging to the same genre (or same singer or group) share similar characteristics, as they are most likely composed of similar instruments (in case of singer or group similar vocal), having similar rhythmic patterns, and similar prosodic distribution [58]. This inspires classification of music emotions where each instance is a music (audio signal) and a label could be genre of the song, such as classical, rock, metal etc. This *music emotions* dataset contains 593 such music instances and each belonging to any of 6 classes [51].

Scene Dataset $Scene$ dataset was created to address the problem of emerging demand for semantic image categorization. Each instance in this dataset is an image that can belong to multiple classes (e.g. Figure 1(a) an image with class label beach and mountain (b) an image with class label field and fall foliage) [3]. This dataset has 2407 images each associated with some of the six available semantic classes (beach, sunset, fall foliage, field, mountain, and urban). Table 7 shows distribution of examples among different class label sets.

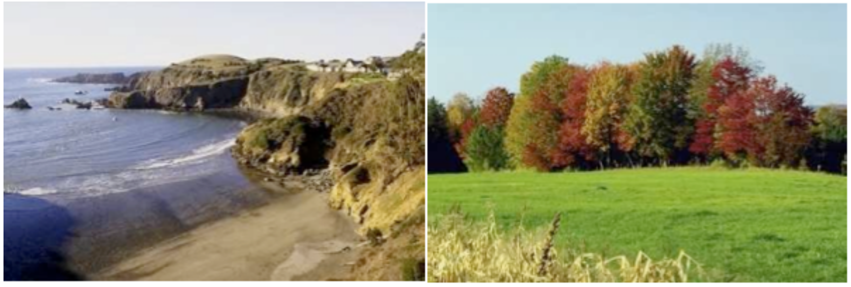
(a) Beach and Mountain (b) Field and Fall Foliage

Figure 2: Example Multi-label Images in Scene Dataset[3]

(Bio)medical Datasets The Computational Medical Center [6] organizes Medical NLP Challenge with a rich set of medical text corpus. This dataset is actually a collection of patient symptom histories, diagnosis and prognoses reports to the insurance companies. *Genbase* [17] and *Yeast* [18] are biological datasets for protein function classification and gene function classification respectively. In *genbase*, each instance is a protein and each label is a protein class (the Prosite documentation ID, i.e. the PDOCxxxxx number) it belongs to. This dataset is small with comparatively large number of labels. In *yeast* data, each instance is a yeast gene described by the concatenation of micro-array expression data and phylogenetic profile. Each of these 2417 genes is associated with one or more of 14 different functional classes. *Elisseeff et. al.* in [18] preprocessed the data and retained the functional classes structured into hierarchies up to 4 levels deep and thats how this data is most commonly used in multi-label problems.

Reuters Corpus Volume (RCV) *Reuters Corpus Volume I (RCV I)* is a large volume text corpus, a collection of over 800,000 newswire stories, collected and manually organized by Reuters Ltd. for research purposes [29]. $Lewis\ et.\ al.$ in [29] made significant efforts to clean this text data with all sorts of text processing techniques, including removing stop words, stemming, transformation to tf-idf, normalization if necessary etc. *(RCV I v2)*. There are three category sets of data: $Topics$ (i.e. major subject of a story), $Industry\ Codes$ (i.e. type of business discussed), and $Regions$ (i. e. geographic locations). Each of these category sets has its own hierarchical or flat structures defined. Its usually common to use different subsets of this data, each containing 6000 data instances on average and with a total number of 101 class labels.
There are also several other Reuters datasets popular in practice such as, $Reuters - 21578$ that has a collection of Reuters financial newswire service in 1987 [28]. After preprocessing this dataset retains 10,788 documents belonging to 90 class labels.

Enron Email Dataset The *Enron Email* dataset contains 517,431 emails (without attachments) from 151 users distributed in 3500 folders, mostly of senior management at the Enron Corp [45]. After preprocessing and careful selection, a substantial small amount of email documents (total 1702) are selected as multi-label data, each email belonging to atleast one of the 53 classes.

Delicious Dataset $Delicious$ dataset, first reported in [53] is a collection of textual data from webpages along with their associated tags. Such web data is collected from a social bookmarking website namely, $del.icio.us$, data collected until April 1, 2007.

EUR-Lex Dataset *EUR-Lex* is a text dataset containing European Union official laws in practice, different kinds of treaties and agreements, parliamentary journals freely available at The EUR-Lex/CELEX (Communitatis Europeae LEX)[9] [34]. This dataset contains 19,348 text documents classified according to three different schemas: $i)$ subject matter (e.g. agriculture), $ii)$ official classification hierarchy called the directory codes (e.g. a document belonging to a class also belongs to all its parent classes), $and\ iii)$ EUROVOC, a multilingual thesaurus maintained by the Office for Official Publications of the European Communities (EUROVOC forms a topic hierarchy). Label

[9] http://eur-lex.europa.eu/en/legis/index.htm

Table 8: Multi-label Datasets Statistics (LCard - Label Cardinality, LDen - Label Density, DL - Distinct Label Sets, PDL - Proportion of Distinct Label)

Data Set	Domain	#Instances	#Distinct Label	LCard	LDen	DL	PDL
Emotions	Music	593	6	1.869	0.311	27	0.046
Genbase	Biology	662	27	1.252	0.046	32	0.048
Enron	Text	1,702	53	3.378	0.064	753	0.442
Scene	Multimedia	2,407	6	1.074	0.179	15	0.006
Yeast	Biology	2,417	14	4.237	0.303	198	0.081
RCV1(avg)	Text	6000	101	2.650	0.026	937	0.156
Delicious	Web Text	16,105	983	19.020	0.019	15,806	0.981
EUR-Lex(Subject Matter)	Text	19,348	201	2.210	1.100	2504	0.129
EUR-Lex(Dir Code)	Text	19,348	410	1.290	0.320	1615	0.083
EUR-Lex(EUROVOC)	Text	19,348	3956	5.310	0.130	16,467	0.851
TMC2007	Text	28,596	22	2.158	0.098	1341	0.047
HiFind	Music	32,978	632	37.3	5.98	-	-
Mediamill	Multimedia	43,907	101	4.376	0.043	6555	0.149

statistics for each of these is reported in Table 8. A processed version of this dataset is available at *http://www.ke.tu-darmstadt.de/resources/eurlex*.

TMC2007 Dataset $TMC2007$ is a collection of flight readiness and discrepancy reports in text [48]. This is a sparse dataset with 28,596 instances and 22 classes. Therefore, feature selection is usually applied to reduce training complexity. One such processed dataset produced using χ^2 features selection resulted in 500 features and is available at: *http://mulan.sourceforge.net/files/tmc2007-500.rar*.

HiFind Dataset $HiFind$ dataset, created and maintained by HiFind Company (a subsidiary of Real Networks) is a large musical metadata [37]. The goal for creating such database was to help making the process of music recommendation system including playlist generation, music retrieval and browsing better efficient. Out of a total of 450,000 music tracks that HiFind categorized over years (1999-2007), a set of randomly selected 32,978 tracks (called the *HiFind* dataset) from 2677 music artists in 39 different genres was first used in [37] for multi-label analysis of music titles. Note that this dataset is very sparse with high label cardinality (37.3). This would also imply redundancy in the data and therefore a significant inter-label dependency.

Mediamill Dataset $Mediamill$ dataset is a multimedia dataset first introduced in a generic video indexing challenge problem that decomposes the problem into unimodal and multimodal video analysis [46]. This dataset contains 85 hours of international broadcast news data, from the TRECVID 2005/2006 benchmark, categorized into 101 class labels.

Webpage Categorization Data The number of web documents is increasing exponentially day by day and that calls for an automatic method for categorization of web documents. This requires a collection of labeled web documents. *Udea et. al.* in [56] used one such web dataset, web pages linked from $yahoo.com$. *Yahoo!* has a hierarchical document class structure with 14 top level categories (e.g. “Arts & Humanities”, “Business & Economy”, “Computers & Internet” etc.) each of which has subcategories and so on.

9 Performance Comparisons

This section compares different *multi-label* algorithms described above based their performance on two benchmark dataset: *scene* and *yeast*. Notice that the analysis in this section is based on the experiments and results described in [52, 20, 69, 14, 68] with a few additional alignment experiments conducted. Table 9 summarizes the algorithms compared and corresponding configurations they are used with. To be consistent, *hamming loss* computed on *scene* and *yeast* dataset for all these algorithms has been considered for evaluation (reported in Figure 3). It is important to note here that

Table 9: Multi-label Algorithms Configurations for Comparison Experiments and Complexity Comments (*n* = number of instances, *t* = number of trees, *a* = avg. number of labels per instance, $k = \lvert\mathcal{L}\rvert$ = number of labels)

Type	**Algorithm**	**Configurations**	**Complexity Comments**
Problem Transform.	LP(SMO)	LP with SMO as the learning algorithm	Can be *exponential* in $\lvert\mathcal{L}\rvert$, additionally base learner complexity
	BR(LR)	BR with Logistic Regression as the learning algorithm	*Linear* in $\lvert\mathcal{L}\rvert$ and additionally base leaner complexity
	BR(SMO)	BR with SMO as the learning algorithm	
	CMLPC	Calibrated Multi-label Perceptrons with pairwise comparisons	Depends on *number of labels per example* times the base learner (can be 2-polynomial)
Algorithm Adaptation	Boostexter	Number of boosting rounds parameter set to 500	$O(nk)$ times weak learner complexity
	Adaboost.MH	Number of boosting rounds parameter set to 50	
Lazy Learning	ML-kNN	Multi-label k Nearest Neighbor with $k = 10$	Can be 2-polynomial in $\lvert\mathcal{L}\rvert$
SVM Based	SVMBinary	SVM^{light} with default parameters as guided in *http://svmlight.joachims.org/*	*Linear* in $\lvert\mathcal{L}\rvert$ and additionally base leaner complexity
	RankSVM	Using higher degree (2-9) polynomials and 1000 iterations	$O(m^2k)$
Ensemble Methods	RAkEL	($k = 4$, $t = 0.6$) and ($k = 4$, $t = 0.7$) for scene and yeast dataset respectively, k = size of labelsets, t= threshold $t =$ 0.6	Exponential in $\lvert\mathcal{L}\rvert$ (usually low)
	LP-RDT	RDT with LP/BR. Total 200 trees constructed, Maximum depth allowed to be the half of the number of attributes, and the minimal instances on a leaf node is 4.	$O(tnlog(n))$
	BR-RDT		$O(t(nlog(n) + an))$
Exploiting Label Dependence	INSDIF	Number of cluster parameter set to be 20% of the size of the training data	Polynominal in k and number of clusters (for SVD)
	EPCC	Ensemble size set to be 10	Exponential in $\lvert\mathcal{L}\rvert$

these two datasets have different properties (Table 8); both datasets are of almost equal size but *scene* dataset has low cardinality (1.074) where *yeast* dataset has reasonably high cardinality (4.237). This means for each instance *yeast* dataset has more complex label space than *scene* dataset, making the former problem more difficult than the latter. This notion is also reflected in Figure 3, *hamming loss* for *yeast* dataset is comparatively higher than *scene* dataset, across all algorithms.

For *scene* dataset *Random Decision Tree* based both methods (i.e. LP-RDT and BR-RDT) ranks best, LP-RDT being slightly better than BR-RDT. More importantly, training time for LP-RDT reported is just 4.06s where the training time for most other algorithms are large, for example RAkEL has a training time of 1995.262s [69]. Such low computation time for RDT based methods is clearly due to simple tree construction approach followed in RDT, specifically almost no computation for selecting a feature at any node. Even though SVM in general had been very effective as a classification algorithm in machine learning, SVMBinary is ranked comparatively poor in order of *hamming loss*. This might be due to the independence assumption among the labels. Calibrated pairwise multi-label learning has a loss value of 0.0969

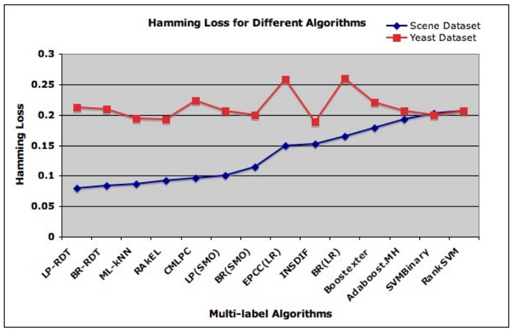

Figure 3: Hamming loss of different multi-label algorithms

which demonstrates the effectiveness of calibration mechanism over ranking by pairwise comparison. INSDIF tries of explicitly model the ambiguity in the input space using a *bag of instances* representation for each instance and this achieves better performance but is outperformed by ensemble based methods such as RDT and RAkEL. This clearly demonstrates the effectiveness of ensembles on top of modeling label correlations.

In case of slightly more sparse *yeast* dataset, INSDIF seemed to outperform all others. This also illustrates the effectiveness of modeling ambiguity in the input space that is core notion in INSDIF. Performance of ensemble based methods such as RAkEL, are also close to the best. In general, ensemble based methods and that captures the dependencies in the data seemed to perform good across both datasets. It is important to note that this comparison presented here is based on *hamming loss* only, and there are other parameters (described in section 7.1.1) need to be explored for any specific application.

Table 9 also reports the computational complexity of each algorithm. Notice that most of the algorithms reported are either exponential or polynomial in $|\mathcal{L}|$. Although in most real problems such complexities are expected to be low due to the real data being less sparse (usually less number of labels, and less number of labels per instance in most applications), they do not really scale very well with large real world applications and especially not suitable for online problems.

10 Future Works and Conclusions

In this paper, an empirical study of different *multi-label* algorithms, their applications and evaluation metrics has been presented. A sparse set of existing algorithms has been organized based on their working principle and a comparative performance analysis has been reported. This study provides useful insights on the relationships among different algorithms and directs light for future research. A few possible future challenges have also been identified: $i)$ while it has been established that exploiting the label correlations is an important factor for improving the performance, this idea in most cases has been used intuitively; a future challenge, therefore, is to theoretically explore the conditional and unconditional dependencies and correlate the performance improvement with each kind of dependence modeling, $ii)$ a number of methods have been quite successful to model small and medium sparse data with reasonably good performances, however, further research attention is needed especially to deal with complicated and large data (notice the significant performance difference reported for *yeast* and *scene* dataset), $iii)$ the computational complexities of most of the algorithms suggest that more efficient algorithms would be needed to achieve scale independence, $iv)$ although it has been established that the data properties such as *label cardinality* can strongly affect the performance of a *multi-label* algorithm, there is no systematic study on how and why the performance varies over different data properties; any such study would be helpful to decide on *multi-label* algorithms for any particular domain, $v)$ with emerging needs for online algorithms, an important future direction would be to design efficient *online multi-label* approaches that scale with large and sparse domains.

References

[1] Arindam Banerjee. Scalable clustering algorithms with balancing constraints. *Data Mining Knowledge Discovery*, 13:2006.

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

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

[4] Klaus Brinker and Eyke Hüllermeier. Case-Based Multilabel Ranking. In *Proceedings of the 20th International Conference on Artificial Intelligence (IJCAI '07)*, pages 702–707, Hyderabad, India, 2007.

[5] Rich Caruana. Multitask learning. In *Machine Learning*, pages 41–75, 1997.

[6] Computational Medical Center. . *Medical NLP Challenge, http://www.computationalmedicine.org/challenge/index.php*.

[7] Nicol Cesa-bianchi, Luca Zaniboni, and Michael Collins. Incremental algorithms for hierarchical classification. In *Journal of Machine Learning Research*, pages 31–54. MIT Press, 2004.

[8] A. Clare and R. D. 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)*, pages 42–53, Freiburg, Germany, 2001.

[9] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[10] Koby Crammer, Michael Kearns, and Jennifer Wortman. Learning from multiple sources. In *In NIPS*, 2007.

[11] Koby Crammer and Yoram Singer. A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, 3:1025–1058, 2003.

[12] F. de Comite, R. Gilleron, and M. Tommasi. Learning Multi-label Alternating Decision Trees from Texts and Data. In *Proceedings of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM 2003)*, pages 35–49, Leipzig, Germany, Julym 2003.

[13] Morris H. De Groot. *Optimal statistical decisions / Morris H. De Groot*. McGraw-Hill, New York :,, 1970.

[14] Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *The 27th International Conference on Machine Learning*, pages 279–286, Haifa, Israel, 2010.

[15] Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. On label dependence in multi-label classification. In *Workshop Proceedings of Learning from Multi-Label Data, The 27th International Conference on Machine Learning*, pages 5–12, Haifa, Israel, June 2010.

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[17] S. Diplaris, G. Tsoumakas, P. Mitkas, and I. Vlahavas. Protein Classification with Multiple Algorithms. In *Proceedings of the 10th Panhellenic Conference on Informatics (PCI 2005)*, pages 448–456, Volos, Greece, November 2005.

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

[19] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27:861–874, June 2006.

[20] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Lozamencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 2008.

[21] N. Ghamrawi and A. Mccallum. Collective Multi-Label Classification. In *Proceedings of the 3005 ACM Conference on Information and Knowledge Management (CIKM '05)*, pages 195–200, Bremen, Germany, 2005.

[22] S. Godbole and S. Sarawagi. Discriminative Methods for Multi-labeled Classification. In *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2004)*, pages 22–30, 2004.

[23] Xian-Sheng Hua and Guo-Jun Qi. Online multi-label active annotation: towards large-scale content-based video search. In *Proceeding of the 16th ACM international conference on Multimedia*, MM '08, pages 141–150, New York, NY, USA, 2008. ACM.

[24] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16-17):1897–1916, November 2008.

[25] Shuiwang Ji, Lei Tang, Shipeng Yu, and Jieping Ye. Extracting shared subspace for multi-label classification. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 381–389, New York, NY, USA, 2008. ACM.

[26] Rong Jin and Zoubin Ghahramani. Learning with multiple labels. In *NIPS*, pages 897–904, 2002.

[27] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97:273–324, December 1997.

[28] D. D. Lewis. Reuters-21578 text categorization test collection distribution 1.0. 2004.

[29] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A New Benchmark Collection for Text Categorization Research. *J. Mach. Learn. Res.*, 5:361–397, 2004.

[30] Eneldo Loza Mencía. Multilabel classification in parallel tasks. In Min-Ling Zhang, Grigorios Tsoumakas, and Zhi-Hua Zhou, editors, *Working Notes of the 2nd International Workshop on Learning from Multi-Label Data at ICML/COLT 2010*, pages 29–36, June 2010.

[31] X. Luo and Zincir A. N. Heywood. Evaluation of Two Systems on Multi-class Multi-label Document Classification. In *Proceedings of the 15th International Symposium on Methodologies for Intelligent Systems*, pages 161–169, 2005.

[32] A. Mccallum. Multi-label text classification with a mixture model trained by EM. In *Proceedings of the AAAI' 99 Workshop on Text Learning*, 1999.

[33] Eneldo Loza Menca and Johannes Frnkranz. Pairwise learning of multilabel classifications with perceptrons. In *In: Proceedings of the IEEE World Congress on Computational Intelligence (WCCI-08), IEEE (2008*.

[34] Eneldo Loza Menca and Johannes Frnkranz. Efficient multilabel classification algorithms for large-scale problems in the legal domain. In *Proceedings of the Language Resources and Evaluation Conference (LREC) Workshop on Semantic Processing of Legal Texts*, 2008.

[35] T. Minka. Expectation propagation for approximate bayesian inference. Ph. D Thesis, MIT Media Labs, 2001.

[36] Ohad Shamir Ofer Dekel. Multiclass-multilabel classification with more classes than examples. In *JMLR Workshop and Conference Proceedings Volume 9: AISTATS*, pages 137–144, 2010.

[37] F. Pachet and P. Roy. Improving Multilabel Analysis of Music Titles: A Large-Scale Validation of the Correction Approach. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(2):335–343, 2009.

[38] Cheong Hee Park and Moonhwi Lee. On applying linear discriminant analysis for multi-labeled problems. *Pattern Recogn. Lett.*, 29:878–887, May 2008.

[39] James Petterson and Tiberio Caetano. Reverse multi-label learning. In J. Lafferty, C. K. I. Williams, R. Zemel, J. Shawe-Taylor, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1903–1911. 2010.

[40] Novi Quadrianto, Alexander Smola, Tiberio Caetano, S.V.N. Vishwanathan, and James Petterson. Multitask learning without label correspondences. In J. Lafferty, C. K. I. Williams, R. Zemel, J. Shawe-Taylor, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1948–1956. 2010.

[41] J. Read. A Pruned Problem Transformation Method for Multi-label classification. In *Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS 2008)*, pages 143–150, 2008.

[42] Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Multi-label Classification Using Ensembles of Pruned Sets. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, volume 0, pages 995–1000, Washington, DC, USA, 2008. IEEE Computer Society.

[43] Gerard Salton. Developments in automatic text retrieval. *Science*, 253:974–979, 1991.

[44] Schapire. Boostexter: a boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[45] Jitesh Shetty and Jafar Adibi. The enron email dataset database schema and brief statistical report, 2004.

[46] Cees G. M. Snoek, Marcel Worring, Jan C. van Gemert, Jan M. Geusebroek, and Arnold W. M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *MULTIMEDIA '06: Proceedings of the 14th annual ACM international conference on Multimedia*, pages 421–430, New York, NY, USA, 2006. ACM.

[47] E. Spyromitros, G. Tsoumakas, and Ioannis Vlahavas. An Empirical Study of Lazy Multilabel Classification Algorithms. In *Proc. 5th Hellenic Conference on Artificial Intelligence (SETN 2008)*, pages 401–406. 2008.

[48] Ashok N. Srivastava and Brett Zane-Ulman. Discovering Recurring Anomalies in Text Reports Regarding Complex Space Systems. In *IEEE Aerospace Conference*, 2005.

[49] Lei Tang, Suju Rajan, and Vijay K. Narayanan. Large scale multi-label classification via metalabeler. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 211–220, New York, NY, USA, 2009. ACM.

[50] F. A. Thabtah, P. Cowling, and Y. Peng. MMAC: A New Multi-class, Multi-label Associative Classification Approach. In *Proceedings of the 4th IEEE International Conference on Data Mining, ICDM '04*, pages 217–224, 2004.

[51] 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.

[52] G. Tsoumakas and I. Katakis. Multi-Label Classification: An Overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

[53] G. Tsoumakas, I. Katakis, and I. Vlahavas. Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, 2008.

[54] 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)*, pages 406–417, Warsaw, Poland, September 2007.

[55] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. *Mining Multi-label Data*. O. Maimon, L. Rokach (Ed.), Springer, 2nd edition, 2010.

[56] N. Ueda and K. Saito. Parametric mixture models for multi-labeled text. In *Advances in Neural Information Processing Systems 15, Neural Information Processing Systems, NIPS*, 2002.

[57] A. Veloso, Jr, M. Goncalves, and M. Zaki. Multi-Label Lazy Associative Classification. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007)*, volume LNAI 4702, pages 605–612, Warsaw, Poland, September 2007. Springer.

[58] Anthony Vetro, Chang Wen Chen, C.C. J. Kuo, Tong Zhang, Qi Tian, and John R. Smith. Content-based audio retrieval: a comparative study of various features and similarity measures. In *Proceedings of SPIE, Vol. 6015 Multimedia Systems and Applications VIII*, 2005.

[59] Hongning Wang, Minlie Huang, and Xiaoyan Zhu. A generative probabilistic model for multi-label classification. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 628–637, Washington, DC, USA, 2008. IEEE Computer Society.

[60] Alicja Wieczorkowska, Piotr Synak, and Zbigniew Raś. Multi-Label Classification of Emotions in Music. pages 307–315. 2006.

[61] Ralf Herbrich Xinhua Zhang, Thore Graepel. Bayesian online learning for multi-label and multi-variate performance measures. In *International Conference on Artificial Intelligence and Statistics, (AISTATS)*, 2010.

[62] Rong Yan, Jelena Tesic, and John R. Smith. Model-shared subspace boosting for multi-label classification. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 834–843, New York, NY, USA, 2007. ACM.

[63] Yiming Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1:67–88, 1999.

[64] Kai Yu, Shipeng Yu, and Volker Tresp. Multi-label Informed Latent Semantic Indexing. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 258–265, Salvador, Brazil, 2005. ACM Press.

[65] 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*, 18(10):1338–1351, 2006.

[66] Min L. Zhang and Zhi H. Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, July 2007.

[67] Min-Ling Zhang and Kun Zhang. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 999–1008, New York, NY, USA, 2010. ACM.

[68] Min-Ling Zhang and Zhi-Hua Zhou. Multi-label learning by instance differentiation. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 1*, pages 669–674. AAAI Press, 2007.

[69] Xiatian Zhang, Quan Yuan, Shiwan Zhao, Wei Fan, Wentao Zheng, and Zhong Wang. Multi-label classification without the multi-label cost. In *SDM*, pages 778–789, 2010.

[70] Zhi H. Zhou and Min L. Zhang. Multi-Instance Multi-Label Learning with Application to Scene Classification. In Bernhard Schölkopf, John C. Platt, and Thomas Hoffman, editors, *NIPS*, pages 1609–1616. MIT Press, 2006.

[71] Shenghuo Zhu, Xiang Ji, Wei Xu, and Yihong Gong. Multi-labelled Classification Using Maximum Entropy Method. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in Information Retrieval*, pages 274–281, 2005.