Multi-label Classification using Ensembles of Pruned Sets

Submitted for Blind Review

Abstract

This paper presents a Pruned Sets method (PS) for multi-label classification. It is centred on the concept of treating sets of labels as single labels. This allows the classification process to inherently take into account correlations between labels. By pruning these sets, PS focuses only on the most important correlations, which reduces complexity and improves accuracy. By combining pruned sets in an ensemble scheme (EPS), new label sets can be formed to adapt to irregular or complex data. The results from experimental evaluation on a variety of multi-label datasets show that [E]PS can achieve better performance and train much faster than other multi-label methods.

1 Introduction

The traditional data mining task of *single-label* classification, also known as *multi-class* classification, assigns each instance d a single label l from a previously known finite set of labels L. A single-label dataset D is composed of n instance-classification examples $(d_0, l_0), (d_1, l_1), \cdots, (d_n, l_n)$. In a *multi-label* classification task, each instance is assigned a *subset* of labels $S \subseteq L$. A multi-label dataset D is therefore composed of n instance-classification examples $(d_0, S_0), (d_1, S_1), \cdots, (d_n, S_n)$.

Although in the past there has been relatively little research involving the multi-label problem, it is in fact quite natural to many domains. For example, in text classification, a news article about US troops in Iraq could intuitively be labelled both `US` and `Iraq`. In addition to text classification [9, 15, 7, 10], multi-label domains include scene classification [17] and genomics [21, 5, 16, 17].

All multi-label problems can be transformed into one or more single-label problems via some *problem transformation* (PT) [17]. In this fashion, any kind of single-label classifier can be used: single-label classifications are made and then transformed back into a multi-label representation. There are many reliable single-label classifiers, all of which can be employed under a PT method for multi-label classification. Some of the most successful PT approaches have worked with Support Vector Machines (SVMs) [7, 8], Naive Bayes [13, 12, 19] and k Nearest Neighbor [21, 11].

It is also possible to modify an existing single-label algorithm for the purpose of multi-label classification. Much of the literature is focussed on modifications to C4.5 trees [5, 16] and AdaBoost [6, 15, 9]. Essentially, these modifications simply employ some form of PT method internally and can usually be generalised to any single-label classifier. Hence all solutions to multi-label classification involve some form of PT method.

There are essentially three fundamental PT methods. They will be referred to in this paper as the Binary Method (BM), the Ranking Method (RM) and the Combination Method (CM).

The most widely used approach, the Binary Method (BM) [17, 7, 8, 21], learns $|L|$ binary classifiers $B_0, \cdots, B_{|L|}$. Each classifier B_j is responsible for predicting the $0/1$ association for each label $l_j \in L$.

Another commonly employed method, the Ranking Method (RM) [17, 15, 11], relies on a single-label classifier giving a probability distribution over all labels. The probabilities define a ranking for the labels. A threshold is used to determine the final subset of labels from this ranking.

Both BM and RM suffer from the *label independence* assumption, and fail to take advantage of any relationships between labels. This means that they both may compose unusually sized multi-label classification subsets or sets whose elements would never co-occur in practice. Performance suffers accordingly.

The Combination Method (CM) [17, 18] creates a single-label problem simply by treating each instance's label combination (or label set) S_i as an atomic label l'_i. For example, the multi-label set $\{a, c, d\}$ would become a single label acd. Hence the set of all distinct label combinations is transformed into a set of possible single labels L' to be considered by the single-label classifier.

CM overcomes the label independence problem, but suffers when many combinations are found infrequently in the dataset. Over many examples, this leads to an overwhelming selection of label combinations which confuses and imbalances the single-label classifier. A second crucial disadvantage is that CM can only assign label combinations to test examples where that combination has been seen in the

training set.

Each PT method varies in terms of CPU and memory requirements. For a dataset D of constant size, BM scales according to $|L|$ as it needs one classifier for each label. Although RM only needs one classifier, this classifier is responsible for assigning probabilities to $|L|$ labels as opposed to two. CM's time complexity is directly proportional to the number of distinct label combinations in the dataset. Each of these combinations becomes a single label during transformation, resulting in potentially very many labels. This is particularly problematic when using SVMs, which are binary classifiers and rely on an expensive pair-wise approach whenever $|L| > 2$.

An ensemble method for multi-label classification was recently pioneered by Tsoumakas and Vlahavas in a system called RAKEL (RAndom K-labEL subsets) [18]. For m iterations of the training data, RAKEL draws a random subset of size k from all labels L and trains a CM classifier using these labels. The authors use SVMs as the internal single-label classifier. A voting process using a threshold t determines the final classification set. Using appropriate values of m, k and t, RAKEL was shown to be better than BM and CM.

The following section presents the PS method. PS is a new method for multi-label classification that addresses some of the limitations of existing methods. It is designed to be fast and to feature low error rates over a wide range of multi-labelling scenarios. In later sections, PS is empirically evaluated and compared with the existing methods just described.

2 Pruning Sets

Consider the graphs in Figure 1. The nodes represent the labels of the *Medical* dataset (statistics of which can be found in Section 3.2) and the edges represent label co-occurrences. In the initial graph, each edge represents one or more co-occurrences between the two nodes (i.e. labels) it connects. In the second graph, edges which represent less than two co-occurrences have been removed. In the third graph, each edge represents at least three label co-occurrences in the dataset. This leaves a simple representation which exposes the core label relationships of the dataset. Note also that this representation still covers 92% of all examples.

The primary motivation behind PS is to capitalise on the most important label relationships found within a multi-label dataset. It focusses on the more frequent label sets while pruning away and breaking up less frequently occurring sets. By pruning much unnecessary and detrimental complexity is avoided. A post-pruning step involving set decomposition ensures the preservation of information from the pruned examples before they are discarded. Pruning is

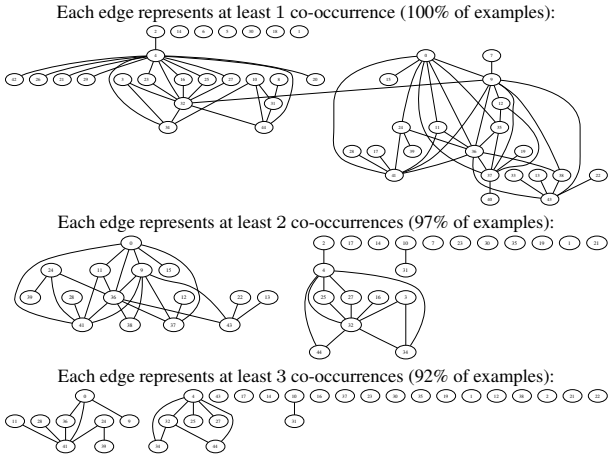

Figure 1. Label co-occurrences in the *Medical* dataset.

controlled by a parameter p which determines how often a label combination must occur for it *not* to be pruned. The PS method consists of the following phases:

Initialisation: D is the multi-label training set. A new empty training set D' is created to hold the final pruned training set. An empty set L' is also created to store label sets with counts of their occurrences in D.

Phase 1. Consider each label set S_i from each training example $(d_i, S_i) \in D$. If (S_i, c) can be found in L' for any count of c, then c is incremented by 1, otherwise a new pair $(S_i, 1)$ is added to L'.

Phase 2. The pruning parameter p is now considered. Pruning is done via exclusion from the set D'. Only training examples $(d_i, S_i) \in D$ where $(S_i, c) \in L'$ for $c > p$ are added directly to D'. The excluded (pruned) examples are passed on to Phase 3.

Phase 3. Information is saved from the training examples which were initially rejected by the pruning parameter. This is done by decomposing each S_i into subsets $s_{i0}, s_{i1}, \cdots, s_{in}$ where $(s_{ij}, c) \in L'$ and $c > p$. These subsets are used to form *new* examples: $(d_i, s_{i0}), (d_i, s_{i1}), \cdots, (d_i, s_{in})$ which are then added to D'. Strategies decomposing each pruned set S_i are discussed below.

Phase 4. Finally a single-label representation is formed from D' using a training procedure like the one used for CM. This preserves the core label relationships in the form of combinations within data upon which any single-label classifier can be employed.

The four phases are marked by numbers in the flow diagram in Figure 2 which outlines the complete PS procedure

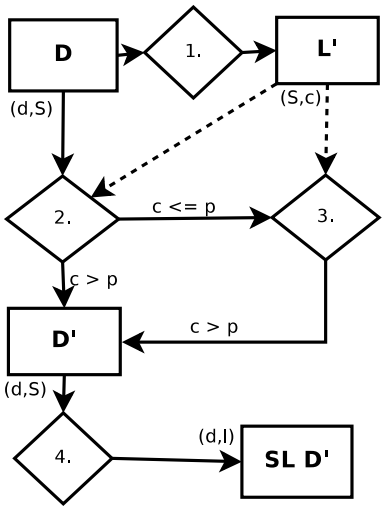

Figure 2. The PS algorithm.

from the initial multi-label training set D to the final single label (SL) representation of the transformed dataset D'.

Phase 3 is an important step to re-introduce information that would otherwise be lost via the pruning procedure (without reintroducing all the complexity). There are different possible strategies to decompose each label set S_i into more frequently occurring sub-subsets. These subsets are then reattached to a copy of the instances they originally belonged to and the resulting examples are included in the final training set.

This process i independent of the pruning parameter p and any internal single-label classifier. The goal is twofold: firstly to preserve as much information as possible about the relationships found within each label set and, secondly, to avoid creating too many small new subsets. Adding many small subsets means a lower average number of labels per instance which in turn causes too few labels to be predicted at classification time. This paper presents two strategies for carrying out this process, each having a parameter b.

Recall that in *Phase 3*, for each $S_i \in D$ we generate every subset $s_{ij} \subset S_i$ where $(s_{ij}, c) \in L'$ and $c > p$.

Strategy A. Rank these subsets firstly by the number of labels they contain and secondly by count c. Keep the *top* b ranked subsets.

Strategy B. Keep *all* subsets *of size greater than* b.

For clarity, examples of the proposed strategies are outlined in Figure 3. Note how all relevant subsets of S_i are discovered, before some of them are discarded, along with S_i. The remaining discovered sets are reattached to a copy of d_i and then added to the final training set.

S_i	$\{l_2, l_5, l_7\}$	*
S_{i1}	$\{l_2, l_5\}$	
S_{i2}	$\{l_5, l_7\}$	
S_{i3}	$\{l_7\}$	
S_{i4}	$\{l_5\}$	*

Strategy A ($b = 3$)

S_i	$\{l_2, l_5, l_7\}$	*
S_{i1}	$\{l_2, l_5\}$	
S_{i2}	$\{l_5, l_7\}$	
S_{i3}	$\{l_7\}$	*
S_{i4}	$\{l_5\}$	*

Strategy B ($b = 1$)

The label sets marked with * are discarded. In this example $S_i = \{l_2, l_5, l_7\}$, $p = 1$ and $L' = \{ (\{l_5, l_7\}, 3), (\{l_2, l_5\}, 4), (\{l_7\}, 3), (\{l_5\}, 2), (\{l_2, l_5, l_7\}, 1) \}$

Figure 3. Examples of Phase 3 strategies A and B.

2.1 Ensembles of Pruned Sets

PS can leverage the label relationships within label sets without considering rare or unusual combinations with other labels. It will function well on many datasets as a standalone method, demonstrating increased speed over existing methods as well as reduced error rates. However, as a standalone method, it still has the limitation of not being able create new multi-label sets which have not been seen in the training data. This presents a problem when working with datasets where labelling is particularly irregular or complex. A method presented by Read [14] can form new combinations using probability distributions in a related fashion to RM. A more general and flexible method (which does not rely on the probability distribution of the single-label classifier) is to combine the results of several classifiers in an ensemble (EPS).

In this paper we propose using PS in an ensemble bagging scheme of m iterations. PS is particularly suited to an ensemble due to its fast build times, and also, because if each PS classifier is built on different subsets of the training set, then different core label combinations can be selected during the pruning process. These can be combined into new predictions via a simple voting procedure at classification time using a threshold t. The ensemble counters any over-fitting effects of the pruning process.

The build phase for the ensemble is detailed in Figure 4. The voting scheme for classification is detailed in Figure 5 and can also be understood from the worked example in Figure 6. Note how, from several core combinations, the process creates a new prediction which may not have been known to any of the five PS classifiers.

3 Experimental Evaluation

In this section the performance of PS is demonstrated in an empirical comparison against all three standard problem transformation methods, as well as the RAKEL algorithm mentioned in Section 1. First we will outline some multi-label evaluation measures, and present a collection of multi-

```
BUILD(D, p, s, m)
 1  ▷ D training set
 2  ▷ p pruning parameter
 3  ▷ s strategy parameter
 4  ▷ m no. of iterations
 5  for i ← 0 to m
 6       do
 7          C_i ← new PS Classifier
 8          D ←randomize D
 9          D_i ← 63.2% of D
10          Train C_i on D_i given parameters p and s
```

Figure 4. The ensemble build phase of EPS.

```
CLASSIFY(d, C_{0...m}, t)
 1  ▷ d test example
 2  ▷ C_{0...m} PS Classifiers
 3  ▷ t threshold
 4  for j ← 0 to |L|
 5       do Y[j] ← 0
 6  for i ← 0 to m
 7       do
 8          y ← Classify d using C_i
 9          for j ← 0 to |L|
10               do
11                  if L[j] ∈ y
12                    then Y[j] ← Y[j] + 1
13  Y ← normalise Y
14  for j ← 0 to |L|
15       do
16          if Y[j] > t
17            then Y[j] ← 1
18            else  Y[j] ← 0
19  return Y
```

Figure 5. The ensemble classification phase of EPS.

$L =$	{	A	B	C	D	E	}
y_{i0}	(	1	0	1	0	0	)
y_{i1}	(	0	0	1	1	0	)
y_{i2}	(	0	0	0	0	1	)
y_{i3}	(	1	0	1	0	0	)
y_{i4}	(	0	0	1	1	0	)
$\sum_{j=0}^{m}$	(	2	0	4	2	1	)
$norm$	(	.22	.00	.44	.22	.11	)
$Y_i =$	(	1	0	1	1	0	)
$Y_i =$	{	A		C	D		}

This is the i^{th} example. $L = \{A, B, C, D, E\}$, $m = 5$, and $t = 0.20$. The prediction bits of the PS classifiers are $y_{i0} \cdots y_{im}$ which are summed for each label and $norm$alised, giving the final classification (under threshold t) of $Y_i = \{A, C, D\}$.

Figure 6. A worked example of the ensemble voting phase of EPS.

label datasets. Then the experimental process is detailed, and the results presented and discussed.

3.1 Evaluation Measures

A multi-label classifier will produce a label subset $Y_i \subseteq L$ as a classification for an instance d_i, which can be compared to the true classification $S_i \subseteq L$ in order to evaluate its performance.

As in single-label evaluation, accuracy can be determined by an "evaluation by example" approach — correctly labelled instances as a percentage of the total number of instances. In the multi-label case, this means that an instance d_i is correct only if $S_i = Y_i$.

For the multi-label situation, there is also the option of "evaluation by label", whereby each label is evaluated separately, and the performance is given as the correctly assigned labels as a percentage of the total number of labels in the dataset ($|L| \times |D|$).

However, the former tends to be overly harsh and the latter overly lenient. Instead, we use accuracy as defined in [17]. Given a classified multi-label test set D:

$$Acc(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|S_i \cap Y_i|}{|S_i \cup Y_i|}$$

We also consider the F_1 measure common to information retrieval. However note that in a multi-label classification context this measure is label-based as opposed to example-based and the average is taken over all examples. Given the classified multi-label dataset D, where p_i and r_i is the precision and recall of the predicted labels Y_i from the true labels S_i for each instance d_i:

$$F_1(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{2 \times p_i \times r_i}{p_i + r_i}$$

Table 1. A collection of multi-label datasets and associated statistics.

	$\|D\|$	$\|L\|$	$LCard(D)$	$PDist(D)$
Scene	2407	6	1.07	0.006
Medical	978	45	1.25	0.096
Yeast	2417	14	4.24	0.082
Enron	1702	53	3.38	0.442
Reuters	6000	103	1.46	0.147

Finally, we consider Hamming loss [17], which is the number of labels in the intersection of Y_i and S_i averaged over all test examples:

$$Hloss(D) = 1 - \frac{1}{|D| \times |L|} \sum_{i=1}^{|D|} |S_i \cap Y_i|$$

Hamming loss is an evaluation by label approach. Note that, contrary to the other two methods, the best possible Hamming loss is 0.0.

3.2 Datasets

There are few benchmark multi-label datasets, however for these experiments we have collected a variety of datasets from different domains. Table 1 displays their associated statistics. *L*abel *Card*inality (LCARD) is a standard measure of "multi-labelled-ness". It is simply the average number of labels relevant to each instance, defined for a dataset D as:

$$LCard(D) = \frac{\sum_{i=1}^{|D|} |S_i|}{|D|}$$

The number of distinct label subsets relative to the total number of examples can be quantified as the *P*roportion of *Dist*inct label combinations:

$$PDist(D) = \frac{|\{S|\exists(d, S) \in D\}|}{|D|}$$

The *Medical* dataset [2] is composed of documents with a brief free-text summary of patient symptom histories and their prognoses which are used to predict insurance codes. The *Yeast* data [18] relates to protein classification. *Scene* [18] relates to the classification of still scenes. *Enron* is a subset of the Enron email corpus [1] labelled by [4]. *Reuters* is a subset of the Reuters RCV1 dataset [10]. The text datasets — *Medical*, *Enron*, and *Reuters* — were all parsed into word frequency vectors and they can be found in their parsed form at [3].

3.3 Experimental Setup

All experiments presented in this paper were carried out using the WEKA [20] framework. In every case SVMs are employed as the single-label classifier. Each method is evaluated by 5×2 fold cross validation (CV) on each dataset.

For consistency, the number of iterations is set to 10 for all ensemble methods. All other parameters are tuned on the training data using internal 5 fold CV, as are the thresholds. Parameters are tuned first and then thresholds secondly in the fashion described below.

EPS finds its optimal parameters using a PS model. RAKEL needs a threshold to run, which is given the initial value of 0.5 (as suggested by its authors) then adjusts it as described below.

During tuning, the values of the parameters were sampled in order of the theoretical complexity they added to each algorithm. For example RAKEL's k parameter was incremented from 2 (the minimum value), whereas the p parameter for PS was decremented from 5. Parameter values were limited to only those which allowed the classifier to build a single model in under one hour (including the internal CV pertinent to that particular value). For example $k = 10$ must be tried under internal $5\times$ CV in less than 1 hour or 10 and all values above will be ignored. The same logic applies to p values.

As detailed in the RAKEL paper, increments of parameter values of k were 2 when $|L| > 14$, and 1 otherwise. The PS method requires a strategy parameter s, denoted by A_b for strategy A and B_b for strategy B. Values of $1, 2, 3$ for b are examined in both cases.

Once parameter values have been selected, thresholds are adjusted. This is also done using $5\times$ CV but, in this case, each fold is tested in a two stage process: the first stage finds the best threshold t to the nearest 0.1, and the second stage finds the best value to the nearest 0.01 within the range $(t - 0.05) \cdots (t + 0.05)$. The average taken over the five folds to produce the final value of t.

It is worth noting that the optimal parameters and thresholds chosen for all algorithms generally tended to be optimal, or close to optimal, within the range of values they were able to test.

All experiments were carried out on AMD Athlon(tm) 64 CPUs at 2.00GHz, with 1 gigabyte of memory.

3.4 Results

Tables 2, 3 and 4 show the results on all datasets for accuracy, F_1 measure and Hamming loss evaluations metrics, respectively. The average mean and the standard deviations for each method over all rounds are shown. Bullet points show significance according to a paired t-test against the CM method which is most relevant to [E]PS and RAKEL. Tables 8 and 9 contain the same results, but provide a direct comparison between RAKEL and [E]PS (instead of CM).

The most frequent parameter configurations and the average thresholds discovered by the tuning phases are pre-

Table 7. Build time (s) for *Reuters*. All $m = 10$ and various p (for [E]PS) and k.

CM	1379				
BM	123				
RM	505				
	p=5	p=4	p=3	p=2	p=1
PS	41	58	80	135	246
EPS	194	277	408	719	1,553
	k=2	k=25	k=50	*k=61**	k=102
RAKEL	10	350	3,627	*22,337*	DNF

*$k = 61$ is the largest value to complete

sented in Table 6.

The average build times (displayed in seconds) are shown in Table 5. These times represent only the time taken to build the complete model for the test data only and do not include the time taken for the parameter and threshold tuning using internal CV (which may or may not have been able to test a full parameter range). Hence in many cases these times represent only a small portion of the total time required for a particular experiment.

In order to fully examine the complexity of exploring the parameter ranges of the methods, all the methods were also timed on a 50/50 train/test split of the *Reuters* dataset. This dataset was chosen specifically due to its high $|D|$ and $|L|$. In this scenario [E]PS and RAKEL were left to try the full range of values for their respective p and k parameters. The methods either completed or ran out of memory (denoted by DNF). A range of results from this experiment is displayed in Table 7.

4 Discussion

Both PS and EPS improve consistently on the standard methods across all measures of evaluation. The improvement is most pronounced on the *Yeast* and *Enron* datasets. Both these datasets have a relatively high label cardinality and non-uniform labelling schemes. This provides more potential for PS methods to excel, as under this kind of data they are especially effective.

As expected, PS performs best in an ensemble scheme, which helps prevent against over-fitting and allows the formation of new label sets. In terms of F_1 measure (Table 3) EPS is statistically superior to CM on all datasets except *Medical* (where the difference is insignificant).

While the ensemble versions of PS tend to perform better, standalone PS still has clear advantages for fast classification. Its error rate is never significantly more than any other method, yet, as we see in Table 5, its build times are a fraction of CM, with which it has most in common. PS also is competitive with the times of the BM and the RM methods which both assume label independence.

An interesting feature of standalone PS is that it performs relatively better in terms of accuracy than in terms of F_1 measure (although this difference is not always statistically significant). This is because PS always prunes away and divides up the most infrequently occurring label sets which also tend to contain the most labels. At classification time, this translates into high precision at the cost of recall and hence the sub-optimal F_1 statistic. However, in many real world scenarios, a consistent emphasis on precision and high accuracy is more important than an optimum trade-off between precision and recall. This trend is avoided under an ensemble scheme, where new combinations are formed and precision and recall can be easily governed by the threshold.

In direct comparison against state-of-the-art RAKEL (Tables 8 and 9) the PS methods also prove superior. Aside from PS's F_1 statistic for the *Yeast* dataset (due to reasons just discussed), all versions of PS are either statistically indistinguishable from RAKEL or superior. This is particularly apparent on *Reuters* where RAKEL clearly failed to find optimal parameters due to the complexity of this dataset.

The complexity of RAKEL is one of its main disadvantages. Although in some cases the average *final* build times in Table 5 shown for RAKEL are less than those for EPS, this is misleading. Unlike EPS, which can tune parameters on a single model (of PS), RAKEL's full ensemble must be built to trial each parameter setting for each fold of internal CV. Parameter tuning for RAKEL is therefore much more expensive and is often terminated prematurely according to the conditions outlined in Section 3.3. In other words, it is computationally expensive and sometimes infeasible for multi-label methods like RAKEL to discover optimal values for their parameters.

In Table 7 RAKEL runs out of memory when $K = 62$ after taking about 6 hours when $K = 61$. PS completes with its most time-expensive p value (1) in about 4 minutes and takes only six times longer in EPS' bagging scheme with 10 iterations. This experiment confirms that RAKEL's k parameter begins to introduce too much complexity for complete operation on larger datasets, whereas the PS algorithms remain feasible. This also explains RAKEL's poor accuracy and F_1 measure on *Reuters*.

Although it may be argued that RAKEL would perform better with greater computing resources than reported here, under such a scenario EPS could also easily increase its number of iterations. Adding iterations adds at most linear complexity whereas, in Table 7, we clearly see that RAKEL's build time increases by a factor of approximately ten each time k is doubled.

As an aside, we further discover from the results that the strategy parameter s of PS is predictable. Table 6 shows

Table 2. Accuracy.

D	CM	BM	RM	PS	EPS	RAKEL
Scene	71.81±1.22	58.28±0.92 ↘	71.72±0.98	71.93±1.08	73.80±0.95	71.58±0.89
Yeast	51.98±0.93	49.64±0.88 ↘	51.95±0.62	52.82±1.30	55.03±0.93 ↗	54.49±0.98 ↗
Medical	74.71±1.32	73.00±1.08	72.71±1.56	74.63±1.51	74.45±2.28	72.55±2.32
Enron	41.02±1.08	38.64±1.05	27.22±0.31 ↘	42.15±0.81	44.09±0.90 ↗	42.98±0.63
Reuters	49.17±0.67	31.91±0.76 ↘	49.08±0.59	49.83±0.59	49.80±0.59	31.80±0.29 ↘

↗, ↘ statistically significant improvement or degradation

Table 3. F_1 measure.

D	CM	BM	RM	PS	EPS	RAKEL
Scene	0.729±0.01	0.671±0.01 ↘	0.724±0.01	0.730±0.01	0.752±0.01 ↗	0.735±0.01
Medical	0.767±0.01	0.791±0.01 ↗	0.743±0.01	0.766±0.02	0.764±0.02	0.784±0.01
Yeast	0.633±0.01	0.630±0.01	0.649±0.01 ↗	0.643±0.01	0.665±0.01 ↗	0.664±0.01 ↗
Enron	0.502±0.01	0.504±0.01	0.335±0.00 ↘	0.520±0.01	0.543±0.01 ↗	0.543±0.01 ↗
Reuters	0.482±0.01	0.421±0.01 ↘	0.485±0.00	0.496±0.00	0.499±0.01 ↗	0.418±0.00 ↘

↗, ↘ statistically significant improvement or degradation

Table 4. Hamming loss.

D	CM	BM	RM	PS	EPS	RAKEL
Scene	0.096±0.004	0.111±0.003 ↘	0.095±0.003	0.095±0.004	0.090±0.003	0.098±0.004
Medical	0.012±0.001	0.011±0.000	0.013±0.001	0.012±0.001	0.013±0.001	0.012±0.001
Yeast	0.213±0.005	0.202±0.005 ↗	0.212±0.009	0.209±0.007	0.211±0.005	0.217±0.008
Enron	0.057±0.001	0.060±0.001	0.055±0.001 ↗	0.055±0.001	0.058±0.001	0.057±0.001
Reuters	0.013±0.000	0.011±0.000 ↗	0.012±0.000 ↗	0.012±0.001	0.013±0.001	0.012±0.000 ↗

↗, ↘ statistically significant improvement or degradation (*N.B. lower is better*)

Table 5. Build time.

D	CM	BM	RM	PS	EPS	RAKEL
Scene	9.8	10.4	3.7	3.8	18.3	9.2
Medical	36.4	7.7	11.9	9.7	51.2	3.4
Yeast	187.8	11.1	34.0	29.8	172.6	64.8
Enron	1565.8	50.9	84.7	59.5	246.1	465.3
Reuters	1379.1	51.7	72.5	176.2	911.9	110.8

Table 6. Parameters and thresholds.

D	CM	BM	RM	PS	EPS	RAKEL
	-	-	t	p, s	p, s, t	k, t
Scene	–	–	0.30	4,A_2	4,A_2,0.37	4,0.30
Medical	–	–	0.10	1,A_2	1,A_2,0.30	8,0.19
Yeast	–	–	0.09	2,$B_{2.5}$	3,B_3,0.07	5,0.20
Enron	–	–	0.10	1,B_2	1,B_2,0.08	10,0.09
Reuters	–	–	0.10	1,A_3	1,A_3,0.21	16,0.06

Table 8. Accuracy: RAKEL and PS methods.

D	RAKEL	PS	EPS
Scene	71.58±0.89	71.93±1.08	73.80±0.95 ↗
Medical	72.55±2.32	74.63±1.51	74.45±2.28
Yeast	54.49±0.98	52.82±1.30	55.03±0.93
Enron	42.98±0.63	42.15±0.81	44.09±0.90
Reuters	31.80±0.29	49.83±0.59 ↗	49.80±0.59 ↗

↗, ↘ statistically significant improvement or degradation

Table 9. F_1 measure: RAKEL and PS methods.

D	RAKEL	PS	EPS
Scene	0.735±0.01	0.730±0.01	0.752±0.01 ↗
Medical	0.784±0.01	0.766±0.02	0.764±0.02
Yeast	0.664±0.01	0.643±0.01 ↘	0.665±0.01
Enron	0.543±0.01	0.520±0.01	0.543±0.01
Reuters	0.418±0.00	0.496±0.00 ↗	0.499±0.01 ↗

↗, ↘ statistically significant improvement or degradation

that strategy A is selected consistently where $LCard(D)$ is low, and B when high (refer also to Table 1).

Although in theory the asymptotic complexity bounds of PS and EPS are not reduced over those of CM or RAKEL, the practical difference cannot be underestimated. Multi-labelled data invariably feature a label distribution conducive to the efficient operation of PS. Multi-labelling schemes are consistently dominated by a small minority of core label relationships (as showed in Figure 1 and Table 1). This assumption can be made despite the exponential number of combinations which are *theoretically* possible with an increasing label set L. The frequency of news articles labelled `Iraq`, `Antarctica`, and `Ireland` would, in practice, be low, if not zero. This explains why PS performs fast despite a theoretical " worst-case" performance similar to other methods.

Memory use is examined in Table 10. It is approximated by the number of instances generated during the problem transformation of a training set D with L possible labels (irrespective of any internal single-label classifier). All values are 'hard' except the Pruning Function $PF(D, p)$ and Decomposition Function $DF(D, s)$ which depend on the distribution of the data in D (and the p and s parameters, respectively). It is guaranteed that $PF(D, p) < |D|$ and that $DF(D, s) < |D| \times LCard(D)$, and also that the complexity of PF is inversely proportional to the complexity of DF. Also, according to the argument concerning the use of PS in practise presented above, PS tends towards logarithmic complexity with respect to p. On *Reuters*, PS actually *reduces* the number of instances in $|D|$ to around 2500 on average whereas RAKEL creates $|D| \times k \times 10$ for k ranging up to $|L|$ (1.82×10^6 instances for $k = 61$). So we observe that PS is efficient in terms of memory as well as speed.

Table 10. Theoretical approximation of memory use.

BM	$\|L\| \times \|D\|$
RM	$\|D\| \times LCard(D)$
CM	$\|D\|$
PS	$PF(D, p) + DF(D, s)$
EPS	$(PF(\subset D, p) + DF(\subset D, s)) \times m$
RAKEL	$\|D\| \times k \times m$

All results indicate that the improvements PS offers are not simply incremental. In many cases the error reduction over other methods is statistically significant, and its performance scales favourably across a wide range of multi-label datasets from different domains including large datasets with thousands of examples and with over a hundred labels.

5 Conclusions and Future Work

This paper introduced a new method for multi-label classification using ensembles of pruned sets. The goal of the pruning procedure is to focus specifically on the core relationships between labels and thereby to reduce the complexity associated with dealing with a large number of distinct or infrequent label sets. The PS method is enhanced by strategies to extract and preserve label relationships from pruned examples without reintroducing the associated complexity.

As a result, the PS method excels in terms of a reduced error rate as well as build time.

While being fully functional as a standalone method, PS is particularly suited to ensembles due to its fast operation and because the randomisation inherent to ensembles counteracts any over-fitting introduced by the pruning phase. Hence PS was also run within an ensemble bagging scheme (EPS) for further reductions to the error rate.

We performed empirical statistical evaluation and the results show that the methods presented in this paper are often superior alternatives to other multi-label methods over a range of multi-labelled datasets. Often the improvements were statistically significant and build times were frequently reduced. The computational and memory complexity were analysed both practically and theoretically. All indications are that the PS methods can be applied effectively and efficiently to many multi-label classification tasks including large and complex multi-label datasets.

References

[1] CALO project: Enron email dataset. URL: `http://www-2.cs.cmu.edu/~enron/`.

[2] Computational medical center: Medical NLP challenge. URL: `http://www.computationalmedicine.org/challenge/index.php`.

[3] Multi-label datasets. URL: `http://www.cs.waikato.ac.nz/~jmr30/datasets.html`.

[4] UC Berkeley enron email analysis project: UC Berkeley enron email analysis. URL: `http://bailando.sims.berkeley.edu/enron_email.html`.

[5] H. Blockeel, L. Schietgat, J. Struyf, A. Clare, and S. Dzeroski. Hierarchical multilabel classification trees for gene function prediction (extended abstract). In *Workshop on Probabilistic Modeling and Machine Learning in Structural and Systems Biology*, Tuusula, Finland, 2006.

[6] Y. Freund and R. Schapire. A short introduction to boosting. *Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[7] S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In *8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2004.

[8] S. Godbole, S. Sarawagi, and S. Chakrabarti. Scaling multiclass support vector machines using inter-class confusion. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 513–518, 2002.

[9] S. Kiritchenko. *Hierarchical Text Categorization and its Application to Bioinformatics*. PhD thesis, Queen's University, Kingston, Canada, 2005.

[10] D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

[11] X. Luo and N. A. Zincir-Heywood. Evaluation of two systems on multi-class multi-label document classification. In *International Syposium on Methodologies for Intelligent Systems*, pages 161–169, 2005.

[12] A. K. McCallum. Multi-label text classification with a mixture model trained by EM. In *Association for the Advancement of Artificial Intelligence workshop on text learning*, 1999.

[13] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[14] J. Read. PPT: A Pruned Problem Transformation method for multi-label classification. In *Proc. of the NZ Computer Science Research Student Conference (NZCSRSC08)*, 2008.

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

[16] J. Struyf, S. Dzeroski, H. Blockeel, and A. Clare. Hierarchical multi-classification with predictive clustering trees in functional genomics. In *Workshop on Computational Methods in Bioinformatics at the 12th Portuguese Conference on Artificial Intelligence*, 2005.

[17] G. Tsoumakas and I. Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3), 2007.

[18] 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)*, 2007.

[19] D. Vilar and M. José. Multi-label text classification using multinomial models. In *España for Natural Language Processing (EsTAL)*, 2004.

[20] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, second edition, 2005.

[21] M.-L. Zhang and Z.-H. Zhou. A k-nearest neighbor based algorithm for multi-label classification. volume 2, pages 718–721. The IEEE Computational Intelligence Society, 2005.