A Pruned Problem Transformation Method for Multi-label Classification

Jesse Read
University of Waikato, Hamilton, New Zealand
jmr30@cs.waikato.ac.nz

ABSTRACT

Multi-label classification has gained significant interest in recent years, paralleled by the increasing use of manual multi-labelling, often known as applying "tags" to documents. Well known examples include Flickr[1], YouTube[2], CiteULike[3] and Google Bookmarks[4]. This paper focuses on Problem Transformation (PT) as an approach to multi-label classification and details these methods as well as their respective advantages and disadvantages. A Pruned Problem Transformation method (PPT) is presented, along with several extensions, designed to overcome such disadvantages. This new method is empirically compared with existing methods, both in terms of accuracy and training time, and the results are encouraging.

Categories and Subject Descriptors

H.2.6 [**Computing Methodologies**]: Artificial Intelligence-Learning[machine learning]; H.3 [**Information Search and Retrieval**]: Information Storage and Retrieval

Keywords

multi-label classification, problem transformation

1. INTRODUCTION

Single-label classification, often known as multi-class classification, is the common machine learning task where a document is assigned a single label l, that is chosen from a previously known finite set of labels L. A dataset D of n documents is composed of document-classification pairs $(d_0, l_0), (d_1, l_i), \cdots, (d_n, l_n)$.

Multi-label classification is an extension of this problem, where documents are classified with a *subset* of labels $S \subseteq L$. A multi–label dataset D of n documents is composed of document-classification pairs $(d_0, S_0), (d_1, S_i), \cdots, (d_n, S_n)$.

[1]http://flickr.com
[2]http://www.youtube.com
[3]http://www.citeulike.org
[4]http://www.google.com/bookmarks

Although in the past there has been relatively little research involving the multi-label problem, it is in fact quite natural to document classification. For example a news article about US troops in Iraq. Given a labelling scheme based on relevant countries, such a document could intuitively be labelled both "US" and "Iraq".

A standard measure of "multi-labeled-ness" is *label cardinality* – $LCard(D)$, which is simply the average number of labels assigned per document. It is defined formally as:

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

The relative number of distinct label subsets assigned throughout a dataset can be quantified by $PDist(D)$; the Probability that a label combination is Distinct within the dataset. This gives some measure of the regularity of the relationships between labels in the dataset. This measure can be defined formally as:

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

A multi-label classifier MC will produce a label subset $Y_i \subseteq L$ as a classification, i.e. $Y_i = MC(d_i)$. Y_i can be compared to the true classification S_i in order to evaluate the performance of the classifier C.

The most common way to carry out multi-label classification is by *Problem Transformation* (PT) to turn multi-labeled training data into a single-label representation, then to use this representation to train one or more single-label classifiers. The single-label output is then transformed back into a multi-label representation via the reverse process. There are many state of the art tried and proven single-label classifiers to choose from which can all be employed under a PT method for multi-label categorisation, some of the most successful PT approaches have worked with Support Vector Machines (SVMs) [4, 3], Naive Bayes [9, 8, 13] and k Nearest Neighbor [15, 7]. SVMs are tempting, especially for text data, due to the high accuracy that they typically demonstrate.

A second way to carry out multi-label classification is to modify an existing single-label algorithm for direct multi-labelling. Such modifications have been done to several single-label algorithms, with most of the literature particularly concentrated on C4.5 [1, 11], and AdaBoost [2, 10, 5, 6]. In reality, most such modifications simply employ a PT method internally and can easily be generalised to any single-label classifier.

There are three fundamental PT methods on which virtually all methods mentioned in the literature are based. For the purpose of simplicity, they can be referred to as PT1, PT2 and PT3.

The most widely documented approach, PT1, learns $|L|$ binary classifiers $B_0, \cdots, B_{|L|}$. Each classifier B_j is responsible for predicting the 0/1 association for each label $l_j \in L$. A 1 indicates that the j^{th} is relevant to a document, and 0 that it is not. Each classifier B_j is trained in a "one-vs-rest" fashion, using documents which are labeled $l_j \in S_i$ as positive examples, against those that are not ($l_i \notin S_i$), as negative examples.

Another commonly employed method, PT2, hinges on the use of posterior probabilities which are output by many single-label classifiers like Naive Bayes. For any document, the single-label classifier can give the probability that it should be assigned each label. I.e. (l_j, λ_j) for each label $l_j \in L$, where λ_j is the probability that l in relevant to the document in question. Ordering these label-probability pairs by λ_j from highest to lowest will produce a ranking of label relevance. The most relevant labels will form the classification subset $Y_i \subseteq L$. They are selected from the top of the ranking with some threshold t; taking each label l_j where $\lambda_j > t$. The threshold is usually selected ad hoc. In regards to the training process, each multi-label document (d_i, S_i) is turned into $|S|$ single-label documents $(d_i, s_{i0}), (d_i, s_{i1}), \cdots, (d_i, s_{i|S|})$.

Both PT1 and PT2 suffer from the *label independence* assumption, and hence fail to take advantage of any relationships between labels. This means that they both may compose classification subsets whose elements would never co-occur in practice. It is also a common occurrence that the subsets created as a classification contain are not a sensible size; containing either far too many or too few labels. Performance suffers accordingly. In the case of PT1, these problems are even further exacerbated by the *class imbalance* problem. There is an overwhelming number of negative examples in each of the individual binary classifiers.

The PT3 method creates its single-label problem simply by treating each document's label subset S_i as a single label l_i. The set of all distinct such labels is used as the set of possible labels L for a single-label classifier to chose from. For example, a document (d, S) where $S = \{a, c, d\}$ is transformed into the single-label representation (d, l) where $l = acd$. The reverse transformation is obvious. The total number of classes the single-label classifier must learn is the total number of distinct label sets found in the training set.

Although PT3 is able to take into account relationships between labels and inherently creates more sensible classification subsets than the other two methods, it suffers when $LCard(D)$ and $PDist(D)$ are high. In this situation there are many rarely occurring combinations throughout the dataset, which forces PT3 to create a huge label set for the single-label classifier to chose from. In practice many label combinations are unique to the training set – often only found in a single document – yet each one will require the creation of separate class for the single-label classifier. The huge selection of such combinations often confuses and unbalances the single-label classifier, leading to significant performance loss. Also, most crucially, PT3 will never be able to assign the correct label combination to a document if that particular combination which was never seen in the training set.

Each PT method varies in terms of CPU and memory requirements. For a dataset D of constant size, PT1 scales according to $|L|$. A separate classifier is trained for each $l_j \in L$. PT2 scales in terms of $LCard(D)$, as there will be $|S_i|$ duplications of the feature set for each multi-labeled training document (d_i, S_i) in D. Unlike PT1 however, PT2 only requires a single classifier. PT3's time complexity is dictated by $PDist(D)$ which indicates the number of classes which are to be generated from the dataset D. This complexity often explodes in problems where the number of such combinations is particularly high. This is especially the case when using SVMs, which are inherently binary, and standard implementations rely on a quadratic pair-wise approach to any single-label problems where $|L| > 2$. In theory, as many classes can be generated as there exist documents in the training set, and even in practice, this number is always greater than $|L|$. Hence PT3 will generally run by far the slowest of all with SVMs.

2. RELATED WORKS

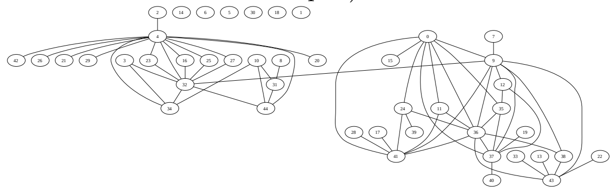
G. Tsoumakas and I. Vlahavas have recently created RAKEL (RAndom K-labEL subsets) [12], a problem transformation method based on PT3. Their method creates m random sets of k label combinations. A single-label classifier (the authors concentrate on SVMs) is built from each of these sets. An ensemble voting process under a threshold t is then employed to arrive at a decision for the final classification set. They show that accuracy can be higher than the standard PT methods. The greatest disadvantage of this approach, acknowledged by the authors, is the time complexity of $O(2^k)$. This makes running the algorithm with many datasets prohibitive. A further disadvantage is the combination of parameters; only certain combinations of $m \times k \times t$ will produce results higher than standard PT3.

3. A PRUNED PROBLEM TRANSFORMATION METHOD

PPT is a simple, but effective and efficient Pruned Problem Transformation method, inspired by the advantages of both the PT3 and PT2 methods, and several aspects found common to most multi-label datasets, particularly text datasets. Consider the graphs in Figure 3, where nodes represent the labels of the *Medical* dataset (refer to Section 4 for more about this dataset), and the edges represent label co-occurrences. When all edges representing a co-occurrences that appear less than three times in the dataset have been removed, this leaves a relatively simple graph which still represents 92% of all documents. PPT concerns itself with this simpler view of a multi-labeled dataset.

The principal concept begins like PT3: Each label subset S_i of each document (d_i, S_i), becomes a single label, and the set of distinct labels becomes the classes for the single-label classifier. PPT differs whenever it deals with a subset

Each edge represents at least 1 co-occurrence (100% of examples):



Each edge represents at least 3 co-occurrences (92% of examples):

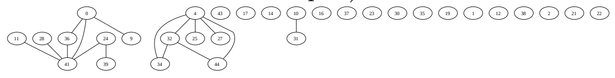


Figure 1: Label co-occurrences in the *Medical* dataset

S_i which occurs in the training data x times or less, where x is supplied as a parameter greater than 0. In this case the subset does *not* become a class value. The most simple option, is to drop all such instances entirely from the training set prior to training. This allows for the creation of a very fast and surprisingly accurate PT method, as the single-label classifier employed by it does not get bogged down by rarely used classes, i.e. it reduces over fitting via a form of Pruning.

However, clearly a lot of information is potentially lost in this process. To save information from a document (d_i, S_i), a technique inspired by the training procedure of PT2 can be employed: Splitting the combination subset S_i, into n sub-subsets $S_{i0}, S_{i1}, \cdots, S_{in}$ where these sub-subsets *do* occur more than x times in the training data. There are a number of ways to chose which subsets to use. In this case we take first take the largest possible subset S_{i1}, then the largest subset S_{i2} which does not contain any elements of S_{i1}, until we are left with an empty set. In other words; subsets where $S_{i1} \cap S_{i2} \cap \cdots \cap S_{in} = \emptyset$. In each case the feature information d_i is duplicated, and each new instance is assigned one of the sub-subsets, i.e. $(d_{i1}, S_{i1}), (d_{i1}, S_{i2}), \cdots, (d_{i1}, S_{in})$, which are all added to the training set (the original (d_i, S_i) is discarded). The pseudo-code of Figure 3 illustrates this process. The process is optional, and in the implementation of PPT it is enabled with a flag "$-n$" (for *n*o information loss), henceforth referred to as PPT$-n$ here. When this option is not specified (PPT standard), these documents will simply be discarded.

Both PPT and PPT$-n$, like PT3, take advantage of relationships between labels in the training data, but are much less prone to over-fitting, and at the same time much more efficient.

However, these methods run into considerable disadvantages when working with more complex than average labelling schemes. When $PDist(D)$ is atypically high, this often signifies that there is little regularity within the labelling, and that there are likely to be many combinations in the test set which may not exist in the training set and thus cannot be accounted for by either classifier.

Whilst authors of RAKEL introduce random subsets to approach this problem, there is also a straightforward method inspired by the classification procedure of PT2 which will

```
1  $X \leftarrow$ distinct subsets that occur $> x$ times in $D_{train}$

PRUNE($D_{train}, (d, S)$)
 1  while $S \neq \emptyset$
 2      do
 3          $T \leftarrow \underset{|T|}{argmax}\, f(T) \in \{T | T \subset S, T \in X\}$
 4          if $T \neq \emptyset$
 5              then $d' \leftarrow$ COPY($d$)
 6                  PUSH ($D_{train}, (d', T)$)
 7                  $S \leftarrow S \setminus T$
 8              else
 9                  then break
10  REMOVE ($D_{train}, (d, T)$)
```

Figure 2: PPT$-n$'s pruning routine, done for each (d, S) where S occurs less than x times in D

not have any effect on build time. PT2 combines top ranking single labels to form a multi-labeled classification, and in a similar way, the PPT implementation can combine top ranking *multi-label combinations* to produce a single multi-labeled classification, and this extension can be applied independent of the training process, as an option that will be referred to here as $-c$. Note that either standard PPT or PPT $-n$ can take on this process. Like PT2, a threshold is necessary to select the top combinations. Figure 3 shows confidence outputs of an example document being combined into a final prediction.

S_k	each label $l_j \in L$								λ_k
0	0	0	1	0	0	1	0	0	0.9
1	0	0	1	0	0	0	0	1	0.8
2	1	0	0	0	0	1	0	1	0.4
3	0	1	0	0	0	0	0	0	0.0
4	1	1	0	0	0	0	0	0	0.0
p	0.4	0.0	1.7	0.0	0.0	1.3	0.0	1.2	$t = 0.5$
Y	0	0	1	0	0	1	0	1	

Figure 3: An example of how PPT$-c$ would use the confidences of several multi-label predictions to form one final multi-label prediction. Each possible multi-label combination S_k is predicted with a confidence of λ_k. l_{jk} indicates the presence of the jth label in the kth distinct subset. $Y_j = 1$ if $p_j > t$, else 0; where $p_j = \sum_{k=0}^{|L|} l_{jk} \times \lambda_k$; $t = 0.5$; $|L| = 8$; |distinct $S| = 5$. The final classification is the set $Y = \{l_2, l_5, l_7\}$

4. EXPERIMENTAL EVALUATION

In this section the performance of the PPT variations are compared against the standard PT methods, and then RAKEL, with various multi-label datasets.

4.1 Evaluation Measures

As in single-label evaluation, accuracy can be determined by an "evaluation by example" approach – correct examples over the total examples ($|D|$) – where in the multi-label case, all labels must be correct for the example to be correct. For multi-labeled output, there is also the option of "evaluation

	$\|D\|$	$\|L\|$	$LCard(D)$	$PDist(D)$
20Newsgroups	19300	20	1.03	0.003
Medical	978	45	1.25	0.096
Articles	3617	101	1.13	0.058
Yeast	2417	14	4.24	0.082
Scene	2407	6	1.07	0.006
OSHUMED	13929	23	1.66	0.082
Enron	1702	53	3.38	0.442

Table 1: A Collection of Multi-label Datasets

by label", where each label is evaluated separately and compared to the total number of labels ($|L| * |D|$), also known as hamming loss. However these two simple measures are rarely used as standalone results for multi-labeled classification, as the former tends to be overly harsh and the latter overly lenient. For the following experiments, accuracy is determined as follows:

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

In other words, accuracy can be obtained for one document by taking the intersection of its true label subset and predicted label subset over the union of these same subsets. The accuracy for the entire test set is then the average of all these accuracies. Other commonly used methods for multi-label evaluation in the literature include precision and recall and the F_1-measure, and hamming loss [12].

4.2 Datasets

Table 1 displays a collection of multi-label datasets, with some important statistics about them. Refer to Section 1 for details of the notation.

Medical[5] includes documents with a brief free-text summary of patient symptom history and their prognosis which are used to predict insurance codes. *20Newsgroups*[6] is the well known 20 Newsgroups dataset. The *Enron*[7] dataset is a subset of the Enron email Corpus[8], labeled with a set of categories developed by the UC Berkeley Enron Email Analysis Project. *Articles*[9] is essentially a website mirror consisting of news and other left-wing articles which are categorised by country, language, and other common news categories such as `Sci/Tech`, and `Economy`. The *OHSUMED*[10] collection is composed of medical abstracts categorised into 23 cardiovascular diseases categories. The *Yeast*[11] data relates to protein classification, and finally, the *Scene*[12] dataset relates to classification of still scenes.

4.3 Results

In Figures 4—10, the variations of PPT are compared with the three standard PT methods PT1, PT2, and PT3. The horizontal axis represents the pruning value x, with a range of $1 - 15$. Higher values represent more pruning. There are two graphs for each dataset, one where vertical axis represents accuracy, and the other where the vertical axis represents build time (in seconds). In each case, SVMs are used as the single-label classifiers. Note that 0.1 is used as the threshold for PT2, as this value shows the best all round accuracy.

The parameter notation $-n$ signifies that information was saved from the documents discarded during pruning. The flag $-c$ indicates that the post-classification voting procedure based on prediction confidences was used to combine multi-label predictions into one. Both these variations to basic PPT are described in Section 3. The $-c$ option requires a threshold and 0.21 is the value used in each of the experiments. The $-c$ option is only used for experiments involving datasets with a high $LCard(D)$ or $PDist(D)$ (refer to table 1). On other datasets the option has no effect on accuracy, and hence is superfluous.

All experiments were carried out on a 1.66GHz Intel(R) CPU with 2 gigabytes of memory running MacOSX. In some cases one or more of the standard PT methods were unable to complete given the resources afforded by this machine or displayed accuracies below the context of the problem and for this reason are absent from two of the graphs.

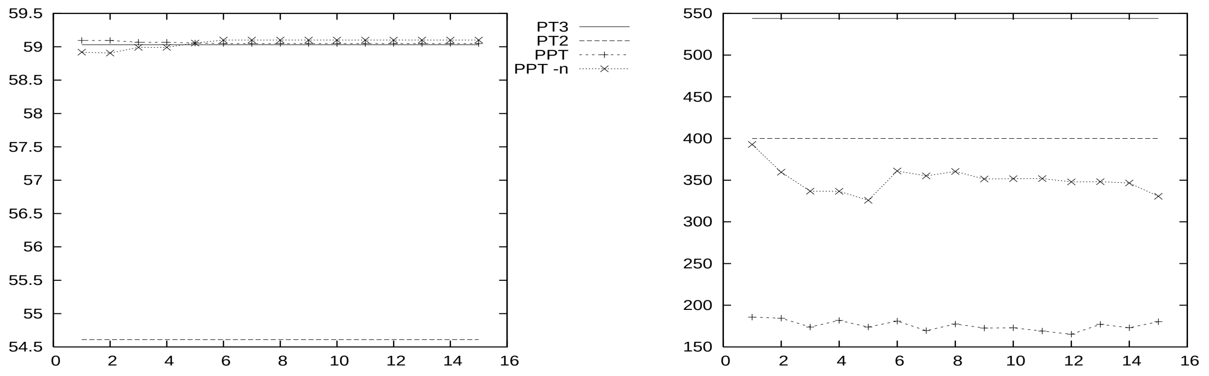
Table 2 compares PPT with the state of the art RAKEL algorithm described in Section 2, using various measures of comparison: Accuracy, the $F1$-measure, hamming loss, as well as build time, and McNemar's test provides the p statistic (from the computed chi-square with one degree of freedom). RAKEL produces different results on each run due to its random subsets, and the figures displayed in the table are an average taken over 10 runs. To determine the parameters to use, combinations of $k \times m \times t$ were run for $k = \{3, 5, 7, \cdots, \frac{|L|}{2}\} \times m = \{5, 10, 15, \cdots, min(|L^k|, 150)\} \times t = 0.5$, which are similar to ranges used by its authors. Each parameter combination was tried once, and the one giving the best result for each dataset was used for 10 runs, the averages of which are displayed in the table. A similar procedure was used to determine a suitable x for PPT, as well as the possible $-n$ and $-c$ flags. Whereas the authors of RAKEL show that $t = 0.5$ generally performs best for their algorithm, PPT works best under approximately $t = 0.21$ and this is used as the default. The rest of the parameters are displayed in the table underneath the respective dataset. Both algorithms have been implemented with the WEKA [14] framework, and the default parameters for SVM is used as the internal single-label classifier.

5. DISCUSSION

PPT invariably improves on all the standard methods for all datasets for at least some parameter combination, usually between the range of $x = 1$ and $x = 6$. Obviously only marginal performance gains can be made over the standard methods when the dataset is barely multi-labeled, e.g. the *20Newsgroups* set, Figure 4. However, while the accuracy gains on these datasets are slight, time complexity is still significantly reduced, particularly in respect to standard PT3.

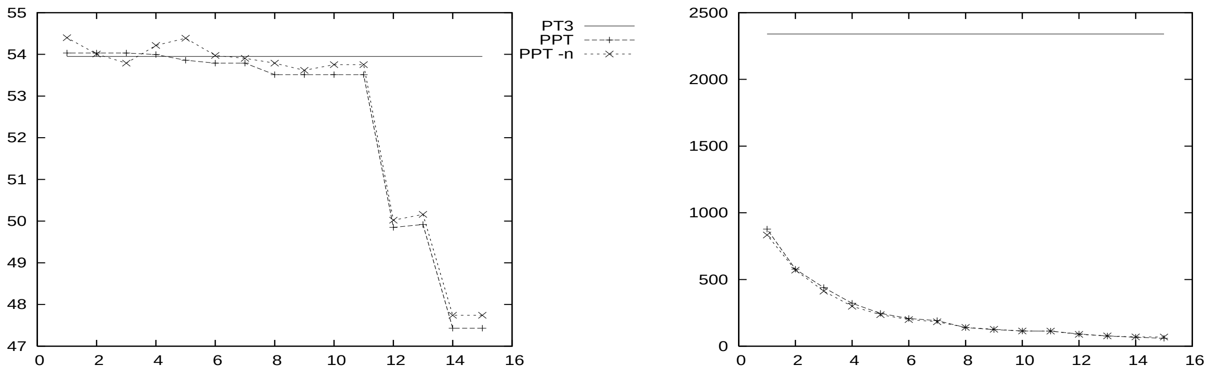
Interestingly, in some cases it was not only faster to simply discard instances, but also more accurate. The experiments with the *Scene*, and in particular the *Yeast* datasets (Figures 8 and 9) show this phenomenon. The *Enron* set (10)

[5]http://ieee-ssci.org/cidm2007/challenge
[6]http://people.csail.mit.edu/jrennie/20Newsgroups/
[7]http://bailando.sims.berkeley.edu/enron_email.html
[8]http://www.cs.cmu.edu/ enron/
[9]http://www.marxist.com
[10]http://dit.unitn.it/ moschitt/corpora.htm
[11]http://mlkd.csd.auth.gr/multilabel.html#Datasets
[12]http://mlkd.csd.auth.gr/multilabel.html#Datasets



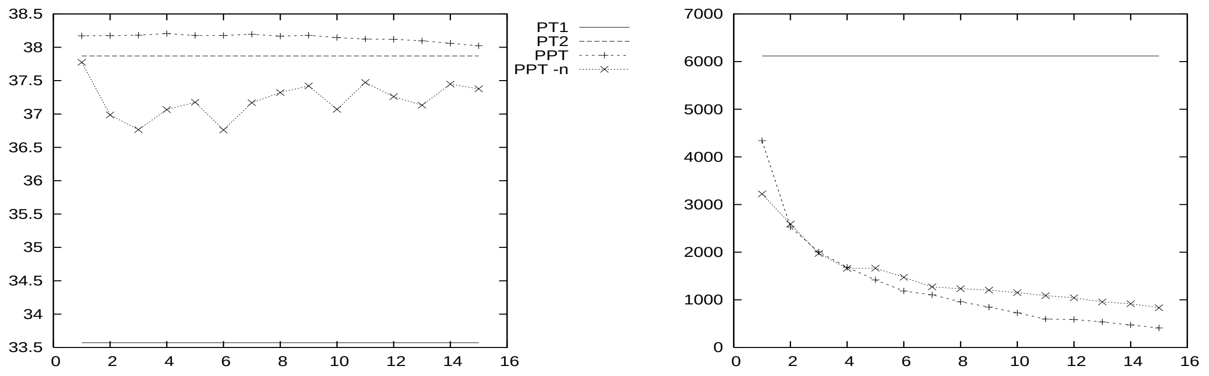
(Accuracy) (Build Time)

Figure 4: ***20Newsgroups*** **Dataset**



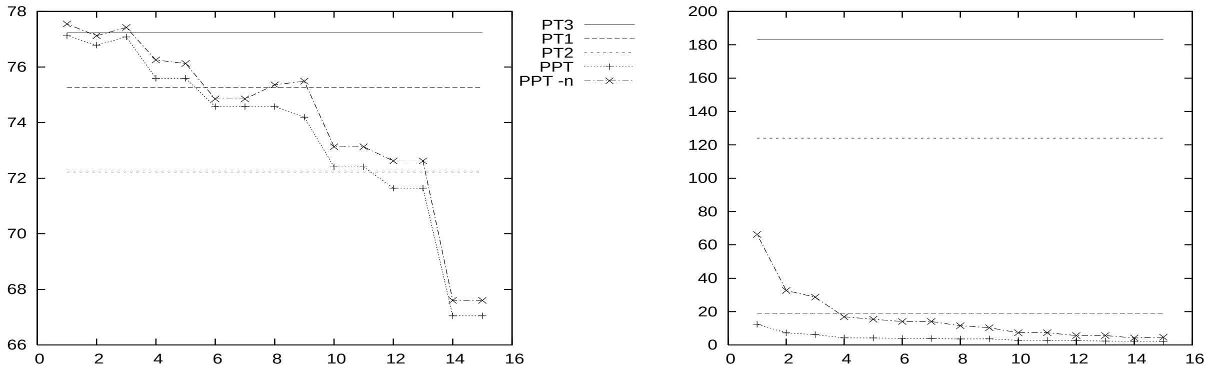
(Accuracy) (Build Time)

Figure 5: ***Articles*** **Dataset**



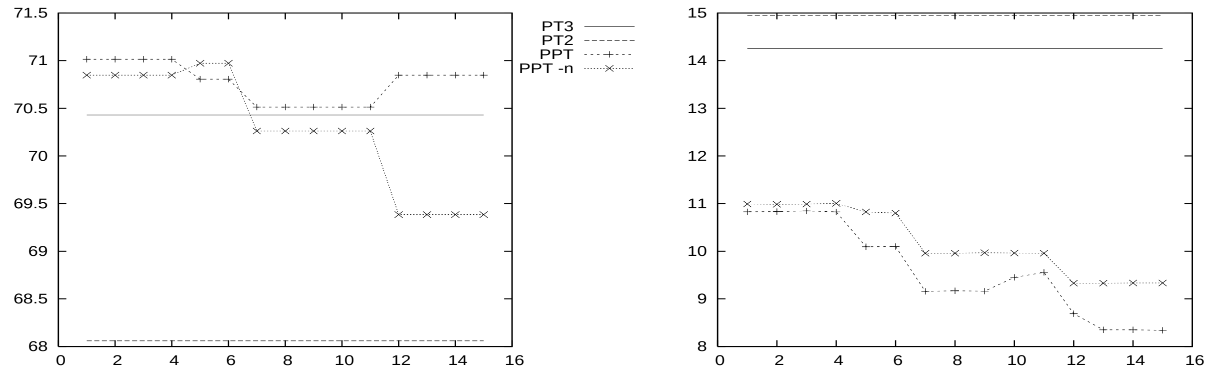
(Accuracy) (Build Time)

Figure 6: ***OHSUMED*** **Dataset**



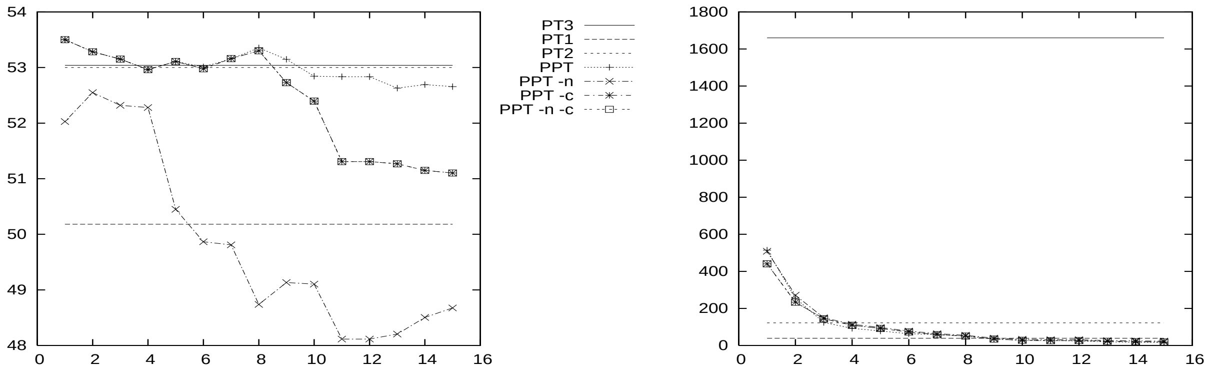
(Accuracy) (Build Time)

Figure 7: ***Medical*** **Dataset**



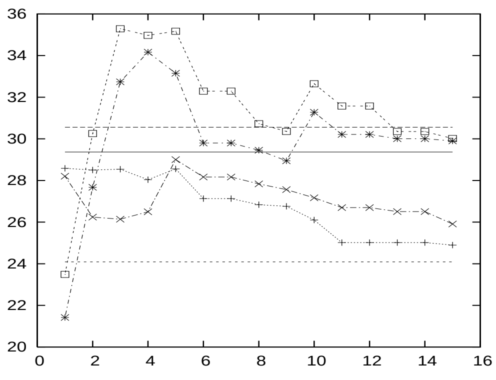
(Accuracy) (Build Time)

Figure 8: ***Scene*** **Dataset**

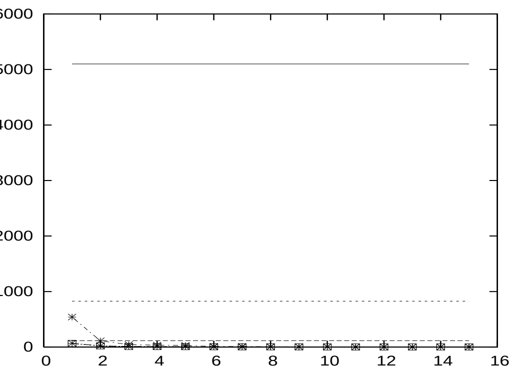


(Accuracy) (Build Time)

Figure 9: ***Yeast*** **Dataset**



(Accuracy)



(Build Time)

Figure 10: ***Enron*** **Dataset**

	Medical		Enron		Scene	
	RAKEL	PPT	RAKEL	PPT	RAKEL	PPT
$F1$	0.776	0.789	0.457	0.503	0.722	0.723
Ham. Loss	0.012	0.011	0.067	0.074	0.102	0.098
Accuracy	0.743	0.776	0.323	0.353	0.696	0.710
McNemar's	$p = 0.295$		$p = 0.000$		$p = 0.077$	
Build Time	190.0	15.0	3162.8	190.0	51.4	4.8
RAKEL par.	$M = 20, K = 27$		$M = 80, K = 21$		$M = 15, K = 4$	
PPT par.	$-n, x = 1$		$-n, -c, x = 5$		$x = 4$	

Table 2: Comparison between RAKEL and PPT

provides the most unpredictable and interesting results, due to its complex nature. Without joining the posteriori confidences, PPT basic suffers against even two of the standard PT methods; PT1 and PT3; yet with the post-classification combination extension ($-c$), a very significant 5 percentage points can be gained over the standard methods. More interesting still, this lead drops when $x < 3$. This also compares to the fact that PT2 (from which the $-c$ extension was inspired) performs better than PT3 – a scenario unique to this dataset.

In Table 2, we see that with the elected parameter combinations, PPT outperforms RAKEL on nearly all measures, and even where the p statistic of McNemar's test does not fall below the 0.05 significance level, the model is quicker to build by at least a factor of 10. PPT continues to distance itself from RAKEL when $|L|$ is greater, as can be seen with the *Enron* dataset, where it excels in both performance and build time.

In nearly all cases the ideal x was found between 1 and 5 inclusive, and for some reason, a threshold of close to $t = 0.21$ proved ideal for all cases where the $-c$ extension was used, thus making ideal parameter configuration reasonably straight forward.

6. CONCLUSIONS AND FUTURE WORK

This paper has introduced a new multi-label Problem Transformation method, PPT, which, under the right parameters, can invariably achieve better classification than the base PT methods which were described, as well as other state of the art methods. PPT focuses on capturing relationships between labels, while pruning and reducing over-fitting. It is able to reduce much of the computation complexity associated with other methods, especially when employing SVMs, while in many cases able to improve on them in terms of accuracy. Its parameters are surprisingly easy to tune, and can be adjusted for optimal accuracy or build time, and an extension to this method, based on posteriori confidences, was also presented, which enables PPT to adapt to particularly complex datasets where otherwise it might suffer.

In the future, related issues worth studying include using different single-labeled classifiers (this paper focuses primarily on the use of SVMs), as well as feature selection, and how this affects the performance of PPT. Particularly of interest and considered for future work is the adaption of PPT to operate in a hierarchical setting.

7. REFERENCES

[1] Dendrik Blockeel, Leander Schietgat, Jan Struyf, Amanda Clare, and Saso 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.

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

[3] S. Godbole, S. Sarawagi, and S. Chakrabarti. Scaling multi-class support vector machines using inter-class confusion. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages

513–518, 2002.

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

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

[6] Svetlana Kiritchenko, Stan Matwin, Richard Nock, and Fazel Famili. Learning and evaluation in the presence of class hierarchies: Application to text categorization. In *Proc. of the 19th Canadian Conference on Artificial Intelligence*, pages 395–406, 2006.

[7] Xiao Luo and Nur 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.

[8] Andrew 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.

[9] Kamal Nigam, Andrew K. Mccallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2/3):103–134, 2000.

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

[11] 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.

[12] 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.

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

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

[15] Min-Ling Zhang and Zhi-Hua Zhou. A k-nearest neighbor based algorithm for multi-label classification. volume 2, pages 718–721 Vol. 2. The IEEE Computational Intelligence Society, 2005.