Department of Computer Science

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

Hamilton, New Zealand

Scalable Multi-label Classification

by

Jesse Read

This thesis is submitted in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy in Computer Science
at The University of Waikato

September 2010

Abstract

Multi-label classification is relevant to many domains, such as text, image and other media, and bioinformatics. Researchers have already noticed that in multi-label data, correlations exist between labels, and a variety of approaches, drawing inspiration from many spheres of machine learning, have been able to model these correlations. However, data sources from the real world are growing ever larger and the multi-label task is particularly sensitive to this due to the complexity associated with multiple labels and the correlations between them. Consequently, many methods do not scale up to large problems.

This thesis deals with scalable multi-label classification: methods which exhibit high predictive performance, but are also able to scale up to larger problems. The first major contribution is the pruned sets method, which is able to model label correlations directly for high predictive performance, but reduces overfitting and complexity over related methods by pruning and subsampling label sets, and can thus scale up to larger datasets. The second major contribution is the classifier chains method, which models correlations with a chain of binary classifiers. The use of binary models allows for scalability to even larger datasets. Pruned sets and classifier chains are robust with respect to both the variety and scale of data that they can deal with, and can be incorporated into other methods. In an ensemble scheme, these methods are able to compete with state-of-the-art methods in terms of predictive

performance as well as scale up to large datasets of hundreds of thousands of training examples.

This thesis also puts a special emphasis on multi-label evaluation; introducing a new evaluation measure and studying threshold calibration. With one of the largest and most varied collections of multi-label datasets in the literature, extensive experimental evaluation shows the advantage of these methods, both in terms of predictive performance, and computational efficiency and scalability.

Acknowledgements

I am extremely thankful to my supervisors, Bernhard Pfahringer and Geoff Holmes.

Bernhard has provided invaluable advice and direction whenever I needed it. He has a talent for spotting technical issues that need to be resolved, and has a constant drive for deeper and more thorough investigations. He has contributed greatly to my knowledge, and I have many times taken advantage of his open door and tireless patience to drop in with a question or to discuss the latest results.

Geoff has always shown his experience in the guidance and insight he has bestowed upon me, particularly at scientific writing. The experience of having him as a supervisor has measurably improved my abilities in this area.

Bernhard and Geoff have both been ideal supervisors. They have been flexible and allowed me to roam—both with ideas, as well as geographically—and at the same time were there to provide support whenever I needed it. They have spent countless hours of their time reading over drafts and providing me with the direction and advice that I needed to finish this thesis. I am also very grateful for them finding me financial support when I needed it.

In addition, I would like to thank Eibe Frank for his suggestions and thorough proof reading, especially regarding 'classifier chains'.

I am also indebted to many people who have contributed in various ways to my research and academic experiences over the past few years outside of New Zealand.

In Ljubljana I thank Marko Grobelnik and Tina Anžič at the Jožef Stefan institute for their support during my stay.

In Leuven I thank Joaquin Vanschoren, Celine Vens, and Hendrik Blockeel, who made my time there productive and rewarding.

In Barcelona many thanks go to Albert Bifet and Ricard Gavaldà. I will never forget their generosity. Beyond the fraternal support he showed me in Barcelona, Albert has become a close friend and invaluable colleague, and I have enjoyed collaborating with him on various projects.

My friends and colleagues here in Waikato also deserve my deepest appreciation: Edmond Zhang, Sam Bartels, Stefan Mutter, Robert Larkens, and others. They have all provided companionship and comradeship during this long journey, both in and out from the computer labs.

I thank my mother, whose unyielding drive and support throughout my childhood has had such profound positive effects on my life.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

It has always been in human nature to collect and classify information. Since the digital revolution, the amount of data has proliferated, and the task of classifying this data manually is rapidly becoming impossible. Automated machine classification will play an ever-more vital role in the future.

Supervised classification is the task of using algorithms that allow computers to learn associations between examples and class *labels*. Supervision comes in the form of previously-labelled examples, from which an algorithm builds a model to automatically predict the labels for new examples. Automated classification involves a variety of domains: text data such as e-mails, web pages, news articles; audio; images and video; medical data; or even annotated genes. Each example is associated with an *attribute* vector which represents data from its domain. Labels represent concepts dependent on the problem domain such as subject categories; genres; gene functions; and other forms of annotation.

Previously-labelled examples are readily available in real world scenarios, usually in the form of human-annotation by an expert. The type of expert depends on the problem domain. In applications such as personal email, or

bookmark collections, users are able to form their own personalised labelling scheme; news articles are commonly filed manually into a variety of categories; doctors can supply medical text classifications; microbiologists can supply examples of gene annotation; and labelled data can even be taken directly from human activity such as computer-network traffic. A supervised classifier trains its model on these examples and continues the labelling task automatically.

In the traditional task of single-label classification each example is associated with a single class label and a classifier learns to associate each new test example with one of these known class labels. When each example may be associated with *multiple* labels, this is known as *multi-label classification*

Although single-label classification is considered the standard task, multi-label classification is by no means less natural or intuitive. The human brain can naturally associate one idea with multiple concepts. A news article about a conference on climate change, for example, can be intuitively labelled both *politics* and *environment*. This type of multiple-association long precedes the digital age.

The popularity of the single-label paradigm most likely originated from the familiarity of dealing with physical objects before the ubiquity of computers and the virtual world. Duplicating physical objects requires extra time, expense, space, and additional complexity for storing, retrieving, and altering them or records of them. This often provides strong impetus for a single-label association. For example, it would be possible to duplicate an article on a conference on climate change in two separate sections of a newspaper. This would make it easier to find the article, but would cre-

ate additional complications: namely the extra physical space on the paper which the duplicated article occupies. For this, and other reasons, it would be typical to place the article in a single section.

In a virtual context, which is evermore relevant in current times, there are no such physical limitations: labels can be associated with data instances without incurring the same physical storage or retrieval costs. Real-world applications have begun to embrace the multi-label paradigm. A good example is how Gmail[1] has replaced the old "folder" metaphor with labels. Many online news sites, for example the BBC[2], often link to the same news article from different category headings, i.e. a multi-label association. A multitude of other sources have also, knowingly or not, embraced the multi-label context. Domains such as microbiology or medicine often inherently require a multi-label scheme: a single gene may influence the production of more than one protein, and a patient's symptoms may be linked to multiple ailments. This explains the explosion of interest in multi-label classification in the academic literature over recent years.

Since the proliferation of data in all domains means that manually assigning labels is becoming infeasible for many applications, automated classification is increasingly seen as an ideal tool.

The multi-label context implies an extra dimension because each example may be associated with multiple labels, as opposed to a single class label. This dimension affects both the learning and evaluation processes.

The evaluation process is no longer straightforward as in traditional single-label learning, since a simple correct/incorrect evaluation no longer suffices

[1] `http://www.gmail.com`
[2] `http://www.bbc.co.uk`

to convey the comparative predictive power of a given classifier. Simple correct/incorrect evaluations *by example* are too harsh by obligating label sets to be exact, and evaluations *by label* are too lenient by ignoring the structure of label sets completely. Thus, different evaluation methods are needed.

Learning is affected by *label correlations*, or *label relationships*, that occur in the multi-label dimension. That is to say that labels co-occur with different frequencies. For example, newspaper articles are more likely to be associated with both category labels `science` and `environment`, than both `environment` and `sport`. In a database of films, the genre labels `family` and `adult` may never occur together throughout the entire collection.

The issue of label correlations directly influences a further issue: *computational complexity*. Instead of choosing a single class label from a label set, a multi-label classifier must consider combinations of labels. Many prior methods for multi-label classification invest considerable computational complexity into modelling label correlations, but as a result become infeasible on even a relatively small scale. As the quantities of data grow ever greater, this situation is exacerbated. Other methods in the literature are based upon very efficient classification paradigms and can scale to large data. However, these methods are often either very domain specific—i.e. specialising within a narrow range of data attributes and dimensions of a specific domain—or are implemented purely to handle large data and are unable to compete with the accuracy of other methods in the literature generally. There is a paucity of methods in the literature which are both efficient enough to handle large datasets as well as able to achieve high accuracy across a variety of data. The now-widespread relevance of multi-label problems implies a need for

such methods.

This thesis advances the field of multi-label classification with *scalable* methods: methods which are able to scale to large datasets, as well as compete with state-of-the-art methods on a wide variety of data sources across different domains.

1.1 Problem Setting and Assumptions

Multi-label classification is a generalisation of single-label classification where each example is associated with a set of labels, as opposed to one label. Therefore the problem settings are similar: a predefined set of labels and a set of training examples associated with these labels. Each example is represented by a vector of attributes. A classifier trains on these examples, and learns to predict the labels for a set of test examples, upon which its performance is evaluated. In multi-label classification, it is usually assumed that the set of labels has at least one element (Godbole and Sarawagi, 2004; Kiritchenko, 2005; Ghamrawi and McCallum, 2005; Zhu et al., 2005), on the basis that if an example has no labels it is irrelevant to the problem context in question. Aside from this exception, an example may be associated with any number of the labels defined in the label set.

The assumptions for this multi-label problem setting are as follows:

1. The set of labels is predefined, meaningful and human-interpretable (albeit by an expert), and all relevant to the problem domain.

2. The number of possible labels is limited in scope, such that they are

human-browsable, and typically not greater than the number of attributes.

3. Each training example is associated with a number of labels from the label set.

4. The number of attributes representing training examples may vary, but there is no need to consider extreme cases of many thousands of attributes since attribute-reduction strategies can be employed in these cases.

5. The number of training examples may be large – a multi-label classifier may have to deal with potentially hundreds of thousands of examples.

Under these assumptions, a multi-label classifier learns from the available training data with the goal of undertaking the task of labelling new data.

1.2 Related Problems

It is important to mention related problems which are not part of the problem setting so as to carefully define the focus of this thesis.

1.2.1 Tagging and keyword assignment

Tagging involves assigning tags to examples in the same way that labels are assigned in the multi-label context. However, tags are usually assigned on the fly, typically in a collaborative process distributed across many users who contribute to the same corpus, often called a *folksonomy*. Hence the set

of tags is dynamic and *not* predefined and therefore does not fit our focus (see Assumption 1). In the past, folksonomies have been criticised for their unreliability and inconsistency in terms of labelling (Chi and Mytkowicz, 2007).

Keyword assignment (or *keyphrase indexing*) with a controlled vocabulary is another task comparable to multi-label learning. The association of keywords with documents is done in a controlled fashion (unlike tags in a folksonomy) but, nevertheless, keywords are usually far too sparse to constitute the label-space of a multi-label problem: the keyword-space is often greater than the attribute-space, or even the number of training examples (see Assumption 2). *Keywords* are usually intended to facilitate content-based search rather than category-based browsing which is appropriate for a labelling context. A thesaurus or external hyper-linked structure such as Wikipedia[3] is often employed to cope with the different task of keyword assignment, as in (Medelyan, 2009). Although there is always possibility for overlap with these related problems, this thesis will not address tagging or keyword assignment directly. As it is a different problem, different approaches are needed.

1.2.2 Hierarchical multi-label classification

The limit on the number of labels in a multi-label problem in terms of human interpretability (see Assumption 1) can be extended considerably by defining an organisational structure in the data, for example a category *hierarchy*, which can facilitate human interpretation of many labels. The *Yahoo!* ontology (Labrou and Finin, 1999) is an example of a huge hierarchy of thousands

[3]http://wikipedia.org

of labels, yet structured for human interpretation and browsing, and thus could be considered a multi-label problem. However, such datasets are rare and the hierarchal arrangement usually facilitates dividing the label set into smaller more manageable subsets which can be treated as separate datasets, as in the treatment of the *Yahoo!* dataset by Tang et al. (2009).

A dataset hierarchy can define important information about the relationships between labels in a specific problem domain. If the hierarchical information is taken into account directly during the classification process, this is known as *hierarchical multi-label classification*. Hierarchical specialisations are beyond the focus of this thesis, although we address hierarchical data generally in Section 2.5 and show why we do not address this specialisation.

1.2.3 Label ranking

Label ranking is the task of creating a model able to *rank*, as opposed to *classify*, the set of predefined labels. For any given instance, a ranking model outputs all the labels in order of their predicted relevance to that particular instance. Although many multi-label classification methods are able to provide a label ranking additionally to a label-set classification, label ranking is nevertheless a separate task requiring separate evaluation, which this thesis does not address specifically.

1.2.4 Other related tasks

Multi-category or *multi-topic* classification is usually a variation of the term multi-label classification, typically dealing with a specific problem domain (where labels can be considered as categories or topics), as is sometimes im-

plied by the terms *subject categorisation* and *document categorisation*. Some variants of the multi-label problem assume the number of labels for each test example is constant or known prior to classification, such as in (Luo and Zincir-Heywood, 2005), however such problems are not frequently found in the real world and this thesis does not consider them.

Multi-instance learning is the task where labels are assigned to *bags* of instances (Maron et al., 1998) as opposed to a single instance. There are a few authors embracing a multi-label multi-instance paradigm (Zhang and Zhou, 2007b; Zhou and Zhang, 2006; Zha et al., 2008) although this is a specific setting not addressed by this thesis.

Multi-task learning involves learning a problem together with other related problems with the aim of creating a better model for the target problem. Because it may allow the learner to use the commonality among the tasks in a similar way to the different labels of a multi-label problem, multi-task learning may overlap in some aspects, but is a different task.

1.3 Approaches and Contributions

This thesis approaches the task of multi-label classification with *problem transformation*, where a multi-label problem is transformed into one or more single-label problems (Tsoumakas and Katakis, 2007). This scheme uses common off-the-shelf single-label classifiers and thus avoids the restrictions of a certain classification paradigm, instead allowing efforts to be placed on improving the classification process itself. This is opposed to *algorithm adaptation*, where a specific classifier is modified to carry out multi-label

classification; often highly suited to specific domains or contexts but not as flexible or as generally applicable as a problem transformation.

The core of this thesis identifies limitations of current problem transformation methods and contributes two significant novel methods: the *pruned sets* method (published in (Read, 2008) and extended in (Read et al., 2008)), and the *classifier chains* method (published in (Read et al., 2009b)). Both these methods put a heavy emphasis on efficiency, but also demonstrate convincing improvements in terms of predictive performance over related methods. We also develop ensemble schemes for both methods. The main contribution of our methods is that they are scalable. This means that they:

- are generally applicable to a wide variety of data sources;
- achieve better predictive performance than other methods in the literature; and
- can scale to large datasets where other high-performing methods do not.

Our claims are justified by theoretical and experimental analysis of time complexity, and extensive empirical comparison to related, well-known and state-of-the-art methods with a large and varied collection of datasets under multiple measures of predictive performance. We compare both to other problem transformation methods as well as to alternative algorithm adaptation methods.

Aside from the major component of novel methods just described, this research also led to general contributions to the field of multi-label classification. These include the following:

- New multi-label datasets, to form one of the largest and varied dataset collections used in the literature
- A detailed investigation of generally applicable threshold-calibration methods
- Novel evaluations involving threshold functions, hierarchical methods, and existing problem transformation methods
- The *log loss* evaluation function for multi-label evaluation
- An *ensemble of binary relevance methods*

Significant parts of the research presented in this thesis have appeared in the following publications.

- Jesse Read, Bernhard Pfahringer, Geoff Holmes, Eibe Frank. *Classifier Chains for Multi-label Classification.* In Proc. of 20th European Conference on Machine Learning (ECML 2009). Bled, Slovenia, September 2009.
- Jesse Read, Bernhard Pfahringer, Geoff Holmes. *Generating Synthetic Multi-label Data Streams.* In Proc. of ECML/PKKD 2009 Workshop on Learning from Multi-label Data (MD'09). Bled, Slovenia, September 2009.
- Jesse Read, Bernhard Pfahringer, Geoff Holmes. *Multi-label Classification using Ensembles of Pruned Sets.* Proc. of IEEE International Conference on Data Mining (ICDM 2008), Pisa, Italy, 2008.

- Jesse Read. *A Pruned Problem Transformation Method for Multi-label Classification.* In Proc. of the NZ Computer Science Research Student Conference, Christchurch, New Zealand (2008).

1.4 Notation

We use the following notation:

- $\mathcal{X} = \mathbb{R}^M$ is the input attribute space
- $\boldsymbol{x} \in \mathcal{X}$ is an *instance*, which can be represented as an M-vector $\boldsymbol{x} = [x_1, \ldots, x_M]$
- $\mathcal{Y} = \{1, \ldots, L\}$ is the set of L possible labels
- Label associations $\boldsymbol{y} \in 2^L$ (i.e. a *label set*[4], subset of $\mathcal{Y}$) can be represented as an L-vector $\boldsymbol{y} = [y_1, \ldots, y_L] = \{0, 1\}^L$ where $y_j = 1$ iff the jth label is relevant (otherwise $y_j = 0$)
- $(\boldsymbol{x}, \boldsymbol{y})$ is an *example* consisting of an instance $\boldsymbol{x}$ and associated labels $\boldsymbol{y}$
- $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$ is a *training set* of N multi-label examples

[4]We use the terms *label associations* or *relevances*, and *label set* interchangeably. Note there is no generally accepted "standard" terminology or formal representation in the multi-label literature. Vector notation and speaking of binary relevances is more convenient when referring to, for example, the "binary relevance" method. On the other hand, speaking of label sets is more convenient in reference to, for example, the "label powerset" and "pruned sets" methods. We show a preference for vector notation, but since this thesis covers such a broad range of multi-label concepts and methods, we use terminology from both the concept of a bit vector, and a label subset.

- When the label reference of a specific example is referred to, i becomes a superscript, such that y_j^i is the binary relevance of the jth label with respect to the ith instance

Logical operations using the $\boldsymbol{y}$-type vector representation are possible:

- $\wedge$ is the AND operation; e.g. $[0,1,0] \wedge [0,1,1] = [0,1,0]$
- $\vee$ is the OR operation; e.g. $[0,1,0] \vee [0,1,1] = [0,1,1]$
- Δ is the XOR operation; e.g. $[0,1,0]\Delta[0,1,1] = [0,0,1]$

For convenience, we can borrow some notation from set theory ($\boldsymbol{y}$ can be thought of as a set where $j \in \boldsymbol{y} \Leftrightarrow y_j = 1$):

- $|\boldsymbol{y}|$ gives the cardinality of $\boldsymbol{y}$, i.e. the number of relevant labels in $\boldsymbol{y}$, where $|\boldsymbol{y}| = \sum_{j=1}^{L} y_j$
- $\boldsymbol{y}' \subset \boldsymbol{y}$ is a label subset, where $|\boldsymbol{y}| > |\boldsymbol{y}'|$ and $\boldsymbol{y}' \wedge \boldsymbol{y} = \boldsymbol{y}'$

We limit the use of the latter notation to Chapter 6, where it is particularly convenient, and prefer to use vector notation elsewhere.

Given training data $\mathcal{D}$, a *multi-label classifier* learns to map the attribute input space to the label output space:

- $h : \mathcal{X} \rightarrow \mathcal{Y}$ is a multi-label classifier
- $\hat{\boldsymbol{y}} = h(\boldsymbol{x})$ is a *multi-label prediction* of classifier h for test instance $\boldsymbol{x}$

The predicted set $\hat{\boldsymbol{y}}$ can then be compared to the *true* classification set $\boldsymbol{y}$ under *multi-label evaluation*. $\mathcal{D}_T$ may denote a set of *test* examples (where

labels are not provided to the classifier). In an evaluation context, we sometimes use $\mathcal{D}$ instead, but at no point do any of our methods use information from test examples prior to or during classification.

Multi-label classification is often conducted in a two-stage process where, upon receiving a test instance, real-valued *confidence outputs* or *scores* are initially provided for all labels and an additional function creates a bi-partition of relevant and irrelevant labels, where the relevant labels become the prediction. In these cases:

- $\hat{\boldsymbol{w}} \in \mathbb{R}^L$ are confidence outputs, where $\hat{w}_j$ is the confidence that the jth label is relevant
- $h : \mathcal{X}^M \to \mathbb{R}^L$ is a classifier that produces confidence outputs
- $f : \mathbb{R}^L \to \{0, 1\}^L$ is a function that turns confidence outputs into a label set prediction (commonly a threshold function)
- $\hat{\boldsymbol{y}} = f(h(\boldsymbol{x}))$ is a multi-label prediction of classifier h with function f, for test instance $\boldsymbol{x}$

1.5 Thesis Organisation

This thesis is structured as follows:

- Chapter 2 provides an in depth study of multi-label data sources and dimensions;
- Chapter 3 discusses multi-label evaluation, and introduces a *log loss* evaluation measure and *threshold calibration method*, and also details

of the setup for experimental evaluation;

- Chapter 4 reviews and defends *problem transformation* methods;
- Chapter 5 provides in depth coverage of prior work in the multi-label literature: both existing work on problem transformation methods as well as alternative approaches;
- Chapter 6 introduces a novel *pruned sets* method;
- Chapter 7 introduces a novel *classifier chains* method. Both Chapters 6 and 7 are largely self-contained in terms of experimental evaluation and discussion, as well as references to related work from Chapter 5; and finally
- Chapter 8 provides a synthesis of the contributions, makes concluding remarks, and discusses future work.

Chapter 2
Multi-labelled Data

In multi-label data, the label space creates an entirely new measurable dimension. This chapter studies that dimension.

Section 2.1 reviews and introduces various metrics for measuring multi-label data. Section 2.2 introduces a collection of multi-label datasets from a variety of real-world domains. Section 2.3 analyses this data in terms of label distribution, and Section 2.4 analyses them in terms of label relationships. Section 2.5 discusses hierarchical data. Section 2.6 briefly reviews attribute selection, extraction and reduction for multi-label data.

2.1 Measuring Multi-labelled Data

As in single-label data, multi-label data can be measured by the number of examples (N), the number of attributes in the input space (M), and the number of labels (L). Here we review measures specific to the multi-label dimension, following the notation from Section 1.4.

*L*abel *Card*inality (LCard) (Equation 2.1) is a standard measure of "multi-labelled-ness", introduced in (Tsoumakas and Katakis, 2007). It is

simply the average number of labels associated with each example.

$$\text{LCard}(\mathcal{D}) = \frac{\sum_{i=1}^{N} |\boldsymbol{y}_i|}{N} \tag{2.1}$$

*L*abel *Dens*ity (LDens) (Equation 2.2), also introduced in (Tsoumakas and Katakis, 2007), relates to LCard, but takes into account the size of the label space.

$$\text{LDens}(\mathcal{D}) = \frac{1}{L}\text{LCard}(\mathcal{D}) \tag{2.2}$$

These measures give a good idea of label frequency, but give no indication of the *regularity* or *uniformity* of the labelling scheme.

We introduce the *P*roportion of *Uniq*ue label combinations (PUniq) (Equation 2.3): the proportion of label sets which are unique across the total number of examples.

$$\text{PUniq}(\mathcal{D}) = \frac{|\{\boldsymbol{y} | \exists! \boldsymbol{x} : (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}\}|}{N} \tag{2.3}$$

This thesis further introduces the *P*roportion of *O*ccurrences of the label set with the *M*aximum frequency (PMax) (Equation 2.4, where $\text{count}(\boldsymbol{y}, \mathcal{D})$ is the frequency that $\boldsymbol{y}$ is found as a label combination in the dataset $\mathcal{D}$). This represents the proportion of examples associated with the most frequently occurring label sets.

$$\text{PMax}(\mathcal{D}) = \max_{\boldsymbol{y} | (\boldsymbol{x}, \boldsymbol{y}) \in D} \frac{\text{count}(\boldsymbol{y}, \mathcal{D})}{N} \tag{2.4}$$

These two measures indicate the level of regularity and uniformity in the

labelling scheme. High $\text{PUniq}(\mathcal{D})$ indicates irregular labelling and, when $\text{PMax}(\mathcal{D})$ is also high, the data exhibits *label skew*. Label skew (often known as the 'power law') is well known to classification, particularly in text data (Gelbukh and Sidorov, 2001). In a multi-label context, label skew translates to a relatively high number of examples associated with the most common label sets, while a relatively high number of examples are associated with infrequent label sets. Label skew is the opposite of label uniformity and, while well known throughout classification literature, is particularly prevalent and exaggerated in the multi-label context (Ráez et al., 2004) where more than one label can be associated with over half of all examples. Label skew becomes *class imbalance* when each label is considered separately as a binary problem.

2.2 Datasets and Applications

Table 2.1 displays our collection of multi-label datasets. All multi-label-specific measurements are as described in Section 2.1. The datasets are ordered roughly by complexity ($N \times L \times M$), with a horizontal line separating larger datasets. Large datasets will be evaluated separately, as detailed in our evaluation setup in Section 3.4.

The rapid increase of popularity of multi-label classification in the recent academic literature has seen with it the emergence of new publicly available datasets, most of which are involved in our collection. However, datasets have still not become nearly as numerous and varied as in the traditional standard single-label literature. To further facilitate the analysis of multi-label data

Table 2.1: A collection of multi-label datasets and associated statistics. n indicates a numeric attribute space; b indicates a binary attribute space.

	N	L	M	LCard	LDens	PDist	PMax	Type
Music	593	6	$72n$	1.87	0.31	0.046	0.137	media
Scene	2407	6	$294n$	1.07	0.18	0.006	0.168	media
Yeast	2417	14	$103n$	4.24	0.30	0.082	0.098	biol.
Genbase	661	27	$1185b$	1.25	0.05	0.048	0.257	biol.
Medical	978	45	$1449b$	1.25	0.03	0.096	0.158	text
Slashdot	3782	22	$1079b$	1.18	0.05	0.041	0.139	text
20ng	19300	20	$1006b$	1.03	0.05	0.003	0.052	text
LangLog	1460	75	$1004b$	1.18	0.02	0.208	0.142	text
Enron	1702	53	$1001b$	3.38	0.06	0.442	0.096	text
Reuters	6000	103	$500n$	1.46	0.01	0.135	0.064	text
TMC2007	28596	22	$500b$	2.16	0.10	0.047	0.087	text
Ohsumed	13929	23	$1002n$	1.66	0.07	0.082	0.084	text
IMDB	120919	28	$1001b$	2.00	0.07	0.037	0.109	text
Bibtex	7395	159	$1836b$	2.40	0.02	0.386	0.064	text
MediaMill	43907	101	$120n$	4.38	0.04	0.149	0.054	media
Delicious	16105	983	$500b$	19.02	0.02	0.981	0.001	text

and the evaluation of multi-label algorithms we have compiled additional datasets from various sources: *Enron* (introduced in (Read, 2008)), *Slashdot* and *IMDB* (introduced in (Read et al., 2009b)), and *LangLog* (introduced here). Let us now review the collection. The relevant sources and citations are given in the text that follows.

Music (Trohidis et al., 2008) is a small dataset concered with labelling instances of music with six possible emotions: `sad-lonely`, `angry-aggressive`, `amazed-surprised`, `relaxing-calm`, `quiet-still`, and `happy-pleased`.

Scene (Boutell et al., 2004) is a relatively small but widely used scene classification dataset involving the following six possible contexts as labels:

`beach`, `sunset`, `field`, `fall-foliage`, `mountain`, and `urban`.

The *Yeast* dataset (Elisseeff and Weston, 2001) is a widely used biological dataset where genes are associated with several of 14 biological functions.

Genbase (Diplaris et al., 2005) is another microbiological dataset concerned with gene-function where, similarly to *Yeast*, each gene can be associated with multiple functions. In this dataset there are 27 labels in the label space.

Medical (Pestian et al., 2007) is a medical-text dataset compiled for the Computational Medicine Centers 2007 Medical Natural Language Processing Challenge[1]. Each document includes a brief free-text summary of patient symptom history and their prognosis, labelled with insurance codes.

We collected the *Slashdot* data from `http://slashdot.org`, with article titles and partial blurbs composing the documents, and subject categories (e.g. `linux`, `technology`, `science`) representing the label space.

We compiled *20ng* from the classic *20 Newsgroups* data (Lang, 2008): a compilation of around 20, 000 posts to 20 newsgroups ranging from `rec.sports` to `politics.guns`, which represent part of a newsgroup hierarchy (where, for example, `rec` and `politics` are internal nodes). Around 1000 posts are available in the data for each of the 20 groups (i.e. labels). This is a somewhat artificial fashion of collection as opposed to other datasets where examples are collected with label sets as they occur naturally. We also note that this dataset is barely multi-label.

We put *Enron* together from a subset of the Enron e-mail corpus[2], already labelled with a hierarchical set of categories developed by the UC Berke-

[1] `http://www.computationalmedicine.org/challenge/`

[2] `http://www.cs.cmu.edu/~enron/`

ley Enron Email Analysis Project[3]. Top level categories are `Coarse genre`, `Included/forwarded information`, `Primary topics` and `Emotional tone`. Each message was labelled by two people, but no claims are made of consistency or comprehensiveness. We considered all the leaf categories as labels.

We compiled the *LangLog* dataset from the *Language Log Forum* (`http://languagelog.ldc.upenn.edu/nll/`) hosted by the University of Pennsylvania, which discusses various topics relating to language (primarily English). We used the 75 topics to represent the label space, which include `language_and_politics`, `errors`, `humor`, and `computational_linguistics`.

Reuters comes from the modern *Reuters RCV1* corpus (Lewis et al., 2004), involving the *Topics* hierarchy. We reduced the attribute space from around 46,000 original numeric attributes to 500 by employing an attribute selection filter for each label, based on information gain, and then taking the top 500 attributes across all labels. This type of attribute selection for multi-label data has been described in (Tsoumakas and Vlahavas, 2007). This dataset is designed around a hierarchy where both leaves and internal nodes can be considered label classification. To flatten this hierarchy, we treat internal nodes (where classifications are possible), as well as the leaves, as the label set.

TMC2007 (Srivastava and Zane-Ulman, 2005) originates from the SIAM Text Mining Workshop[4] and contains instances of aviation safety reports that document problems that occurred during certain flights. The labels represent the problems being described by these reports. We use a reduced version of this dataset with the top 500 attributes selected, as specified in (Tsoumakas

[3] `http://bailando.sims.berkeley.edu/enron_email.html`
[4] `http://www.cs.utk.edu/tmw07/`

and Vlahavas, 2007).

The *Ohsumed* collection (Hersh et al., 1994) is a subset[5] of the MEDLINE database[6]. This database consists of peer-reviewed medical articles, labelled with disease categories.

We gathered *IMDB* from the Internet Movie DataBase[7] (from `http://www.imdb.com/interfaces#plain`). We used the movie plot text summaries as examples labelled with their relevant genres. Note that this dataset is a larger updated version from that which we used in (Read et al., 2009b).

MediaMill (Snoek et al., 2006) originates from the 2005 NIST TRECVID challenge dataset[8] which contains annotated video data. The label space is represented by 101 "annotation concepts" such as `Explosion`, `Aircraft`, `Face`, `Truck`, `Urban`.

The *Delicious* dataset was collected and preprocessed by Tsoumakas et al. (2008) from the *del.ico.us* social bookmarking site[9]. This dataset actually exceeds the scope for multi-label problems that we set in Section 1.1 (see assumption 2). It is, in fact, a modified tagging problem: the label space was not predefined prior to labelling and the size of the label space is greater than the size of the input space ($L > M$). However, we include it in our collection to help demonstrate the scalability of the algorithms we present and analyse in later chapters.

These datasets and further information about them can be obtained at the sources referenced in the above text. Datasets which we compiled or mod-

[5] `http://www.mat.unical.it/Olex-GA/examples.htm`
[6] `http://www.ncbi.nlm.nih.gov/entrez/query/static/overview.html`
[7] `http://imdb.org`
[8] `http://www.science.uva.nl/research/mediamill/challenge/`
[9] `http://del.ico.us`

ified are available at `http://meka.sourceforge.net`. Many of the datasets in our collection are part of the growing collection maintained by the Machine Learning and Knowledge Discovery Group in the Aristotle University of Thessaloniki, available at `http://mlkd.csd.auth.gr/multilabel.html#Datasets`.

The number of publicly available multi-labelled datasets has grown considerably, but still by no means represents an exhaustive list of possible applications for multi-label classification, and we expect the number of available datasets to continue increasing.

2.3 Label Distribution

Label distribution refers to the distribution of frequencies at which label sets occur within the data. This may be characterised by the measures described in Section 2.1.

LCard varies considerably across datasets, ranging from close to 1.0 (e.g. *20ng* and *Scene*) where most examples are associated with only a single label, to more than 4.0 (e.g. *Yeast*, *MediaMill*). LDens is usually very low, i.e. labelling is usually very sparse, and *Yeast* and *Music* are the exceptions where nearly 30 percent of the label space is associated on average with each example (in the latter case, this is probably more on account of its small label space—see also *Scene*).

Low label cardinality (and density) is typical of text and media classification, where most examples fit naturally under a single label scheme, and multi-labelling has been introduced to resolve ambiguities. Consider the

Label	Top Level Category
Attachment(s)	Included/forwarded information
Newsletters	Included/forwarded information
Forwarded Email(s)	Included/forwarded information
Legal Advice	Primary Topics
Company Image – Current	Primary Topics
Internal Company Policy	Primary Topics
Humor	Emotional Tone
Admiration	Emotional Tone
Triumph/Gloating	Emotional Tone
Worry/Anxiety	Emotional Tone
...	...

Figure 2.1: A subset of the label space of the *Enron* dataset.

Scene image-classification dataset, where most images are naturally relevant to only a single label, such as `mountain`, `field`, or `sunset`. Multiple labels are used to resolve occasional ambiguities, for example when `mountain` and `field` are both relevant to one particular image.

High label cardinality is often observed in datasets with a very narrow domain. Examples include biological datasets, like *Yeast*, where genes are expected to have multiple functions, and text datasets like the *Enron* email dataset which is specific to the Enron Corporation[10], where label categories take the form of a checklist. Figure 2.1 provides a small sample of the label space of the *Enron* dataset to illustrate this characteristic.

In our collection, the label cardinality of all datasets we consider as multi-label problems is less than 5.0. We have already discussed the exception: *Delicious*, where $L > M$, and thus the problem is better treated as a tagging or keyword-assignment problem, as discussed in Section 1.2.1.

[10] `http://en.wikipedia.org/wiki/Enron`

Various label distributions are displayed in Figure 2.2, as the overall composition of label set frequencies. These distributions can be approximated by a Poisson distribution:

$$POISS(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

with λ as a function of label cardinality and k as a function of the size of the label space ($\lambda = \text{LCard}(\mathcal{D}), k = L$). Only the highly structured labelling scheme of the *Yeast* dataset deviates significantly from this distribution (where label-set sizes of 2, 4, and 6 are all more common than sizes 3 and 5). Figure 2.2 shows how datasets contain very few examples that are associated with more than about 10, even when the dataset involves over 100 possible labels (see Table 2.1).

The variance in $\text{PUniq}(\mathcal{D})$ and $\text{PMax}(\mathcal{D})$ values indicate the different levels of regularity of the labelling that may be found in real-world data. *Enron* and *LangLog* are examples of irregular labelling, given the large proportion of unique label sets (high $\text{PUniq}(\mathcal{D})$), and the latter is also quite skewed given the relatively high number of examples associated with the most frequent label set (high $\text{PMax}(\mathcal{D})$). *20ng* is a much different dataset: it is hardly multi-label at all, and (partly as a consequence of this) only 0.3 percent of label sets are unique in the dataset—i.e. labelling is very regular—partially due to the artificial way examples were originally collected (mentioned in Section 2.2). *IMDB* is an example of a dataset with relatively few unique label sets, but where the frequency of the most commonly occurring label set is relatively high. *Delicious* demonstrates the difference between the dimensions of a tagging problem and a typical multi-label problem.

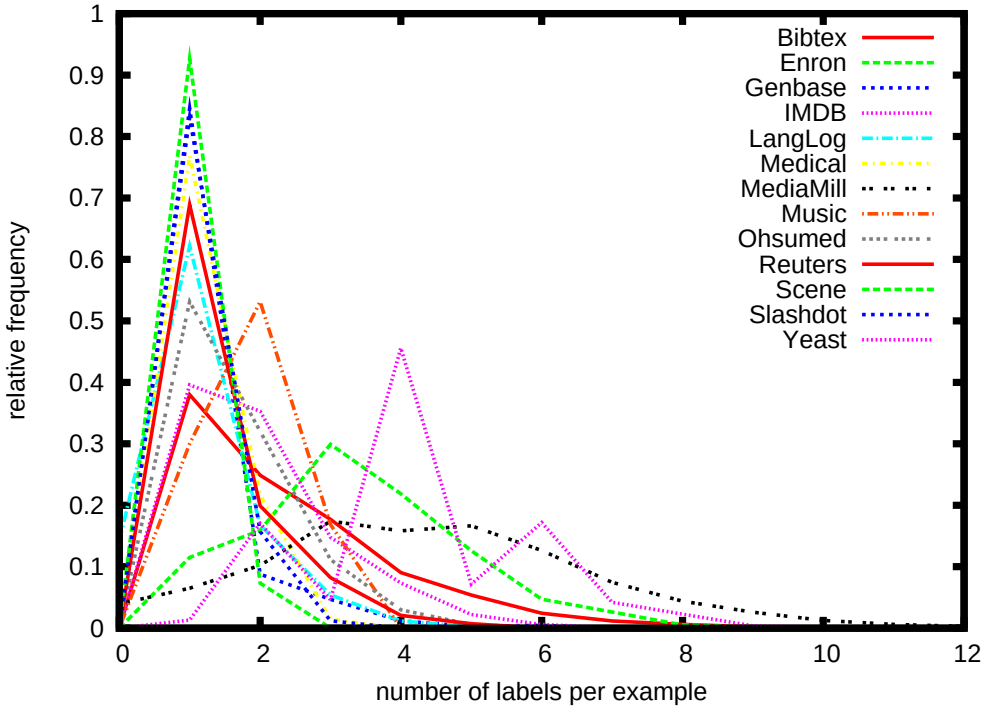


Figure 2.2: The label distributions of various datasets.

We have just looked at label distribution, which deals with label-set frequencies. In the following section we discuss relationships which exist between labels themselves.

2.4 Label Relationships

In all multi-label problems relationships exist between labels. In the absence of label relationships multi-label data is uninteresting, since each label could be assumed independent and treated as a separate binary problem without any loss of information, and the point of multi-label classification would be lost. The presence of label relationships is considered throughout the literature (Ji et al., 2008; Godbole and Sarawagi, 2004; Tsoumakas and Vlahavas, 2007; Read et al., 2008; Sun et al., 2008; Yan et al., 2007; Loza Mencía and Fürnkranz, 2008), and is worth analysing in detail.

We can say that two labels can be measured to occur together with a certain probability i.e. the conditional dependence of one label on another. The conditional probability of the jth label being relevant given that the kth label is relevant, is $P(y_j|y_k)$. $P(y_j)$ is the prior probability. Note that prior probabilities in multi-label classification sum to more than 1.0, since labels are not all mutually exclusive, and that $\text{LC}\textsc{ard}(\mathcal{D})/N = \sum_j^L P(Y_j)$. We can say that there is some "relationship" between labels y_j and y_k when $P(y_j|y_k) \not\approx P(y_j)$.

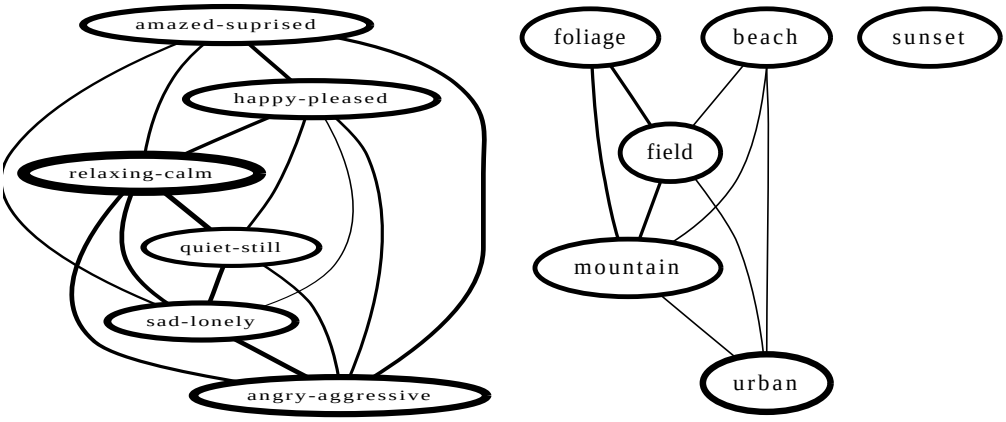
We can visualise label relationships in different ways. Figure 2.3 shows a graph of co-occurrences probabilities for the labels of various datasets. Figure 2.4 shows conditional and prior probabilities in the form of heatmaps, which

Table 2.2: Existing label combinations and their respective frequencies for labels `Comedy`, `Short`, `Animation` and `Family` in the *IMDB* dataset.

Combination	Freq.
`Comedy`	6802
`Comedy,Short`	3219
`Short`	3213
`Animation,Comedy,Short,Family`	1469
`Animation,Short`	1086
`Family`	643
`Animation,Comedy,Short`	435
`Comedy,Family`	302
`Animation`	291
`Animation,Family`	182
`Animation,Comedy`	128
`Short,Family`	118
`Animation,Comedy,Family`	108
`Animation,Short Family`	89
`Comedy,Short,Family`	79

provide a more compact representation. We see "columns" in the heatmap around labels with high priors (i.e. $P(y_j)$ values), most clearly in *Enron* and *MediaMill* due to their high label cardinality. The remainder of noticeable effects (where $P(y_j|y_k) \not\approx P(y_j)$) are strong domain-dependent relationships, seen most clearly in *Genbase* and *Medical*.

These graphs only display binary correlations. Often label relationships are best analysed as *combinations*. Table 2.2 represents a subset of four labels from the *IMDB* dataset (`Animation`, `Comedy`, `Short`, and `Family`), and the combinations in which they occur most frequently. Note that, for example, the combination `Animation,Comedy,Short,Family` occurs much more often in the data that just the combination `Comedy,Short,Family`; illustrating that label relationships can be more complex than just binary correlations.

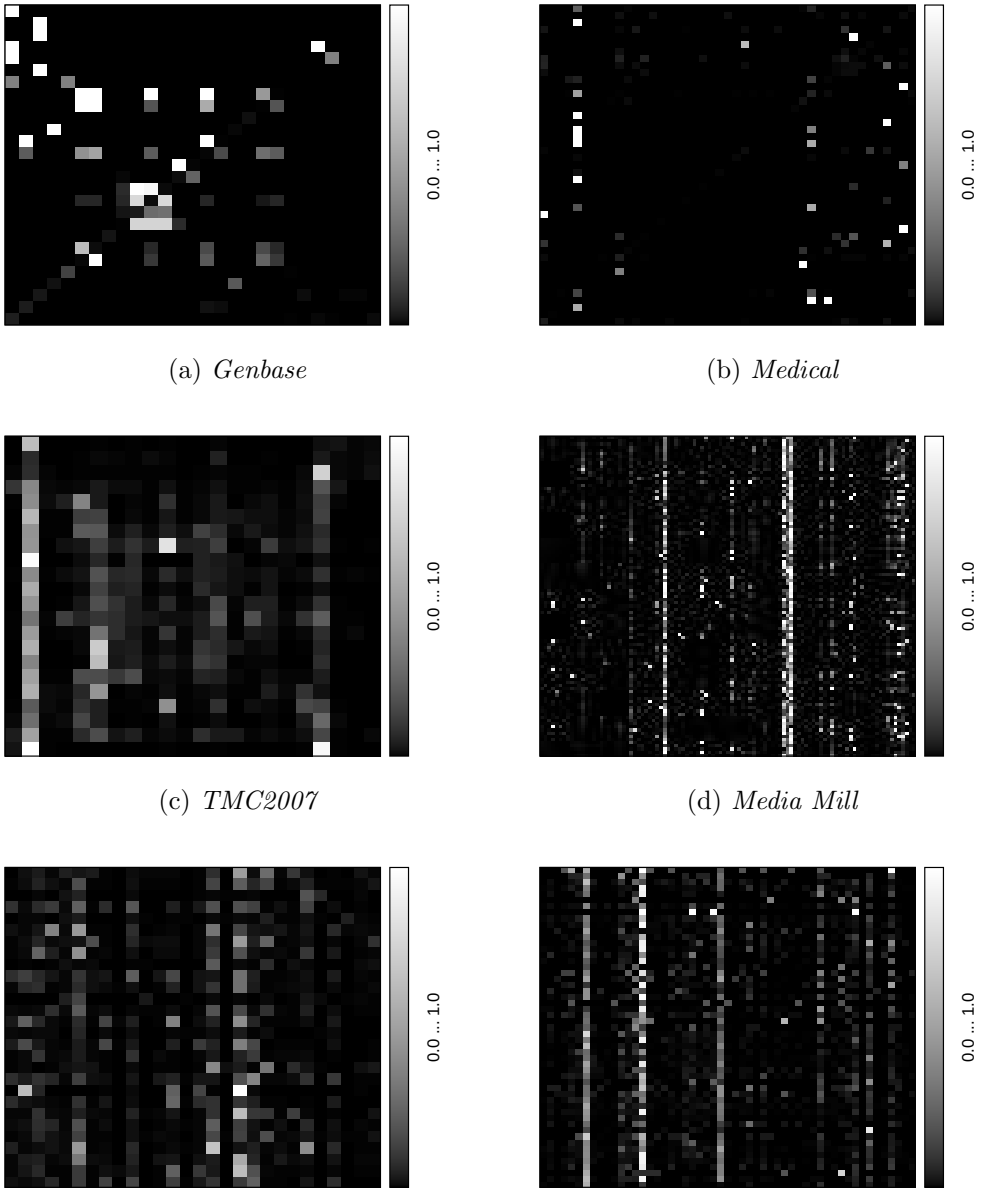


(a) *Music* (b) *Scene*

(c) *Genbase* (d) *Medical*

(e) *Yeast* (f) *IMDB*: a subset of 7 of 28 labels

Figure 2.3: Label relationships. Node thickness indicates prior probability $P(y_j)$; edge thickness indicates co-occurrence probability $P(y_j \wedge y_k)$. Note that only binary relationships can be interpreted from the graph (two nodes connected via a third may not necessarily co-occur in the data).



(a) *Genbase* (b) *Medical*

(c) *TMC2007* (d) *Media Mill*

(e) *IMDB* (f) *Enron*

Figure 2.4: Heatmaps representing an $L \times L$ matrix $\boldsymbol{M}$ of conditional probabilities such that $\boldsymbol{M}_{jk} = P(y_j|y_k) = P(y_j \wedge y_k)/P(y_k)$ where subscript j is a row and k a column. The diagonal (bottom left to top right) holds the prior probabilities, i.e. $\boldsymbol{M}_{jj} = P(y_j)$, most clearly seen in *Genbase*.

In text classification, the instance space of an example labelled *both* `A` and `B` can represent a mixture of `A`-examples and `B`-examples. A newspaper article labelled `economy` and `war` may be composed of words relating to both labels. However, this does not necessarily apply to all domains. An image containing `beach`-images and `people`-images may not be a mix of typical `beach` and `people`, since people generally look somewhat different on a beach (as opposed to a city street for example), and beaches can also look different when they are covered in people. The relationships between attributes and labels in multi-label domains have not yet been fully unravelled or explored on a large scale in the literature, although in (Read et al., 2010) we conduct an investigation which showed some of these effects.

2.5 Hierarchical Data

In some cases, the relationships between labels may be explicitly structured in the dataset in the form of a hierarchy. Consider the hierarchy of *20ng* in Figure 2.5[11]. More expansive and intricate hierarchies also exist, such as the *Reuters RCV1* hierarchy (mentioned in Section 2.2), and also particularly in genomics data, where a DAG (directly acyclic graph) structure may also be present; for example the gene ontology (*GO*) (Ashburner et al., 2000), functional catalogue (*FunCat*) (Ruepp et al., 2004) and *MIPS* (Mewes et al., 1997) hierarchies. When a hierarchical structure is taken into account directly by a classification process, this is *hierarchical classification*. Hierarchical classification does not necessarily imply multi-label classification, but

[11]Note that we made small changes to the hierarchy; namely combining branches `soc`→`religion`→ and `talk`→`religion`→ into a single branch: `religion`→

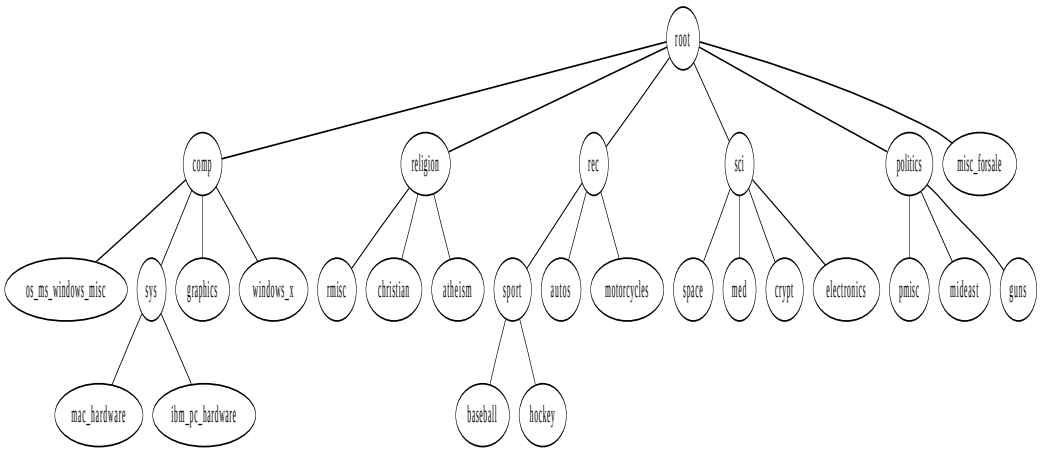


Figure 2.5: The *20ng* hierarchy.

the two contexts often overlap.

Hierarchical classification can either be carried out *internally* or *externally* to the classification process (or a combination of both approaches). In *internal* hierarchical classification, a 'hierarchical' classifier models the dataset-defined hierarchy internally. If the hierarchical structure is not modelled, this is simply regular '*flat*' classification. In *external* hierarchical classification, which is most commonly used on subject and category hierarchies, regular 'flat' multi-label classifiers are arranged in alignment with the hierarchy: an internal node is a classifier; the branches from it are classifications; the leaf nodes reached are label predictions. Classifications filter down from a root node (or root nodes) to the leaves, and all predicted leaf nodes represent a multi-label prediction. For example, in Figure 2.5, eight classifiers would be used to predict up to 20 possible labels. The classifier at each internal node is, by default, ignorant of the hierarchy and only makes local predictions. A special case is where internal nodes are also considered as possible label

predictions (Cesa-Bianchi et al., 2006; Kiritchenko, 2005).

We must ask the question: is there a general advantage to hierarchical classification over a flattened classification problem? For external hierarchical classification, in terms of predictive performance, much of the literature indicates that there generally is not. This is due to *error propagation*, where errors are propagated down a hierarchy (Cai, 2008; Godbole et al., 2002) and label-correlation information is lost as classifications proceed downward toward the leaves: see in Figure 2.5 that any associations between `religion` and `politics`, and `comp` and `sci`, are already lost at the first branch despite the fact that these pairs of labels are likely to be related to each other. The accuracy of a naive Bayes method was shown by Godbole et al. (2002) to be similar in both flat and hierarchical contexts. Similar findings were made by McCallum et al. (1998).

In terms of internal hierarchical classification, a small number of methods have benefited from the presence of a hierarchy in certain domains; predominantly microbiology, where expansive and complex ontologies are often involved. In this context, Vens et al. (2008) implemented hierarchically-adapted decision-tree classifiers, and report good results. However, such a specific approach falls outside the focus of this thesis of generally-applicable methods.

Even external hierarchical approaches are specific approaches in the sense that they rely on the presence of a hierarchy. It is, however, possible to use unsupervised methods to construct *artificial hierarchies*: general methods for generating a hierarchy from the data, rather than relying on the presence and quality of a dataset-defined hierarchy. Cai (2008) proposed an approach

based on agglomerative clustering. Tsoumakas et al. (2008) used a modified k-means clustering algorithm to create hierarchical structure where there is none previously defined. Artificial hierarchies can facilitate the interpretation of results and reduce computational complexity in many multi-label contexts by providing each classifier (at each internal node) with data with a reduced label space (Tsoumakas et al., 2008). On the other hand, the number of overall classifiers required in such a scheme may contribute significant overhead in terms of running time and, in many practical scenarios, may undermine the gains. Improvement in predictive performance under artificial hierarchies in terms of multi-label classification was largely unsubstantiated, so we conducted an investigation of our own.

Tables 2.3 and 2.4 display the results (in terms of predictive performance, and run time, respectively) of a small experiment using the MULAN framework (Tsoumakas et al., 2009b) comparing regular flat classification to two types of artificial hierarchies: *random* and *clustered*, as introduced in (Tsoumakas et al., 2008). The multi-label-specific *accuracy* evaluation measure is reviewed in Chapter 3, and the `BR` and `LC` methods in Chapter 4. The results show that hierarchies generally offer little benefit to classification. The top-performing algorithm in terms of predictive performance was in all but two cases one of the standard flat methods. Although on the larger datasets using a hierarchy does provide considerable relief in terms of running time, the overhead of a hierarchy caused significant memory problems, resulting in did not finish (DNF)—as defined in Section 3.4—on *IMDB* and *Delicious*.

To summarise: hierarchical classification is beneficial in too few circum-

Table 2.3: Predictive performance (in terms of multi-label *accuracy* – see Chapter 3) for methods `BR` and `LC` (see Chapter 4) in both flat and hierarchical classification contexts. Top performing method set in **bold** for each dataset. Naive Bayes is the base classifier.

			hierarchical `LC`		hierarchical `BR`	
	flat `BR`	flat `LC`	clustered	random	clustered	random
Music	0.425	**0.519**	0.415	0.437	0.428	0.448
Scene	0.430	**0.603**	0.511	0.524	0.446	0.543
Genbase	**0.601**	0.340	0.348	0.450	0.390	0.460
Yeast	0.409	**0.464**	0.430	0.391	0.404	0.392
Medical	0.531	**0.663**	0.561	0.252	0.568	0.506
Slashdot	0.411	**0.486**	0.316	0.386	0.341	0.383
20ng	0.309	**0.578**	0.318	0.387	0.283	0.309
LangLog	0.010	0.098	0.080	**0.103**	0.043	0.053
Enron	0.170	**0.313**	0.289	0.189	0.197	0.154
Reuters	0.098	**0.278**	0.155	0.221	0.202	0.270
TMC2007	0.363	**0.525**	0.384	0.425	0.410	0.415
Ohsumed	0.229	**0.367**	0.290	0.303	0.317	0.324
IMDB	0.170	**0.137**	DNF	DNF	DNF	DNF
Bibtex	0.178	0.198	0.213	0.221	**0.228**	0.210
MediaMill	0.022	**0.250**	0.243	0.101	0.260	0.143
Delicious	**0.06**4	0.023	DNF	0.031	DNF	DNF

stances to be considered in depth in this thesis, which approaches multi-label classification in a more general sense. Internal hierarchical classification is beneficial only in specific domains, whereas general approaches, such as external hierarchical classification under artificially generated hierarchies, give poor results in terms of predictive performance. However we note that any multi-label method can be considered in an external hierarchical context, and thus excluding this approach from the focus of this thesis does not exclude it from future application to methods presented within.

Table 2.4: Running time (in seconds) for methods `BR` and `LC` in both flat and hierarchical classification contexts. Fastest method set in **bold** for each dataset. Naive Bayes is the base classifier.

			hierarchical `LC`		hierarchical `BR`	
	flat `BR`	flat `LC`	clustered	random	clustered	random
Music	0.59	**0.06**	1.19	0.86	1.15	0.99
Scene	2.48	**2.20**	3.28	2.83	3.96	3.42
Genbase	3.10	**1.12**	6.29	3.62	6.67	4.17
Yeast	**2.22**	5.85	4.05	3.37	4.43	4.14
Medical	55.06	19.47	16.10	**10.21**	19.26	15.58
Slashdot	81.15	105.65	32.31	**24.88**	39.20	28.25
20ng	792.55	195.90	166.89	**88.53**	250.51	133.04
LangLog	143.26	50.92	28.36	**12.40**	33.97	24.72
Enron	102.78	142.58	29.23	**22.31**	28.83	23.50
Reuters	259.85	320.07	102.80	**37.59**	109.42	46.14
TMC2007	48.89	269.59	96.15	**45.66**	96.42	46.71
Ohsumed	664.82	2192.28	132.39	**100.72**	154.74	117.58
IMDB	**8264.12**	70248.63	DNF	DNF	DNF	DNF
Bibtex	598.35	1000.75	269.53	**118.97**	269.97	126.22
MediaMill	537.01	2642.50	887.03	**169.97**	930.68	222.29
Delicious	**1377.72**	1515.09	DNF	1768.87	DNF	DNF

2.6 Attribute Preprocessing

Preprocessing in the attribute space (or feature space) of multi-label data can be done in the same way as in other facets of classification, such as single-label classification and unsupervised classification (i.e. clustering), where attribute vectors are used to represent each example. The type of attribute space of the datasets in our collection was indicated in Table 2.1 as either *b*inary or *n*umeric.

The number and kind of attributes in the attribute space is determined through *attribute selection*, which is generally divided into *wrapper* and *filter* approaches. A wrapper approach selects attributes based on their performance in classification, whereas a filter approach selects attributes based solely on dataset information. The former is generally too slow for large datasets, so the latter is much more commonly used (Kiritchenko, 2005; Mladenic and Grobelnik, 1998).

We encoded word information for all our gathered text datasets using the ubiquitous "bag of words" approach (Lewis, 1998), thus creating a sparse vector model where $\mathcal{X} \in \{0,1\}^M$, such that the ath element of $\boldsymbol{x}_i \in \mathcal{X}$ represents the binary presence of the ath word in the ith document. We account for the top 1000 or so occurring words. The number of attributes can be very large, particularly in text data, in which case the number of attributes can be reduced with further attribute selection. In selecting the best attributes in a multi-label context, it is possible to select attributes in a label-based fashion (by ranking the attributes with respect to each label, and then taking the top k overall), or in a label-set-based fashion (by considering each label set as a single class label, and carrying out attribute selection as

in any single-label problem). Tsoumakas and Vlahavas (2007) employed the former technique on *TMC2007* as did we with the *Reuters* data. Kiritchenko (2005) reviews attribute selection methods in detail.

In the hierarchical context, a distinction exists between *local* and *global* attribute selection (Kiritchenko, 2005), which is a concept parallel to internal and external hierarchical classification. Global attribute selection involves a single process for extracting attributes from the dataset (the same as in a regular flat context) whereas local attribute selection employs a separate filtering process at each internal node of a hierarchy, so that the attribute space is more general near the root and increasingly specific nearer the leaf nodes where sister nodes are more closely related. Note that hierarchical attribute selection can be carried out even when hierarchical classification itself is not used. There is a wealth of works that investigate different forms of hierarchical attribute selection (Bade et al., 2006; Wibowo and Williams, 2002; McCallum et al., 1998; Wang et al., 1999; Zhang et al., 2009).

Other preprocessing strategies include encoding the relationships between labels and attributes (Ghamrawi and McCallum, 2005; Luo and Zincir-Heywood, 2005; McCallum et al., 1998), as opposed to doing so at classification time. In (Read et al., 2009a) we uncovered and outlined in detail these relationships, but this thesis places instead its focus on the task of classification.

Chapter 3
Multi-label Evaluation

In the single-label context, predictive performance is easily handled under the traditional *accuracy* measure, where each test example is either correct or incorrect, and performance is given by the number of correctly classified test instances relative to the total number of test instances. In the multi-label space, predictive performance can be measured in two ways: either as *label-set based evaluation* where each label set (i.e. each example) is evaluated separately, or *label-based evaluation* where the binary relevance of each individual label is evaluated separately. If the traditional accuracy measure is applied in a label-set based fashion then each label set prediction must match exactly, which can be too harsh, since even a single false positive or false negative label makes the example incorrect. On the other hand, if each label is evaluated as a separate example, this measure tends to be overly lenient due to the typical sparsity of labels in multi-label data—essentially creating *class-imbalance*—where predicting ultra-conservatively is heavily rewarded. As a result, additional measures of evaluation are needed.

Section 3.1 reviews existing evaluation measures specific to multi-label classification; Section 3.2 discusses the issue of thresholding functions and

takes a fresh look at threshold calibration as a part of multi-label evaluation. Section 3.3 presents a new multi-label *log-loss* measure. Section 3.4 synthesises these sections in the details of the multi-label setup for the experimental evaluations in later chapters.

3.1 Multi-label Evaluation Measures

First we consider the basic measures of evaluation just mentioned: label-set-based 'accuracy', which is referred to in the literature as *exact match* (Equation 3.1) and label-based 'accuracy', measured as *Hamming loss* (Equation 3.2, where $\hat{\boldsymbol{y}}_i \Delta \boldsymbol{y}_i$ is the symmetrical difference between $\hat{\boldsymbol{y}}_i$ and $\boldsymbol{y}_i$ (the logical XOR operation). These represent example-based and label-based accuracy, respectively.

$$\textsc{Exact-match}(\mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} 1_{\hat{\boldsymbol{y}}_i = \boldsymbol{y}_i} \tag{3.1}$$

$$\textsc{Hamming-loss}(\mathcal{D}) = \frac{1}{NL} \sum_{i=1}^{N} |\hat{\boldsymbol{y}}_i \Delta \boldsymbol{y}_i| \tag{3.2}$$

A multi-label measure of accuracy was introduced in (Godbole and Sarawagi, 2004) (Equation 3.3). This is the ratio of the size of the union and intersection of the predicted and actual label sets (represented by the logical AND and OR operations in bit-vector notation, respectively), taken for each example, and averaged over the number of examples. This measure has been used often in the multi-label literature (Godbole and Sarawagi, 2004; Tsoumakas and Katakis, 2007; Read et al., 2008). This thesis refers to this measure

throughout as simply ACCURACY in a multi-label context.

$$\textsc{Accuracy}(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}\frac{|\boldsymbol{y}_i \wedge \hat{\boldsymbol{y}}_i|}{|\boldsymbol{y}_i \vee \hat{\boldsymbol{y}}_i|} \tag{3.3}$$

The *F-measure*, commonly used in information retrieval, has also been popular in multi-label classification (Tsoumakas and Katakis, 2007; Spyromitros et al., 2008; Read et al., 2008). For any vector of label associations $\boldsymbol{z} \in \{0,1\}^T$, a label is relevant if $z_j = 1$ and predicted if $\hat{z}_j = 1$ (in a corresponding vector of *predicted* label associations), and from this we can define:

- *precision* as the fraction of predicted relevances which are actually relevant—$(|\boldsymbol{z} \wedge \hat{\boldsymbol{z}}|)/|\hat{\boldsymbol{z}}|$; and
- *recall* as the fraction of actual relevances which are also predicted—$(|\boldsymbol{z} \wedge \hat{\boldsymbol{z}}|)/|\boldsymbol{z}|$).

F-measure (F1) is calculated as:

$$\text{F1}(\hat{\boldsymbol{z}}, \boldsymbol{z}) = \frac{2.0 \times \mathit{precision} \times \mathit{recall}}{\mathit{precision} + \mathit{recall}}$$

We have specified a T-vector $\boldsymbol{z}$ instead of an L-vector $\boldsymbol{y}$, because in the multi-label context there are several ways to average this measure, namely:

- *micro-averaged*; one vector $\boldsymbol{z} \equiv [y_j^i, \ldots, y_L^N]$ (of NL values)
- *macro-averaged (by label)*; L vectors of $\boldsymbol{z}_j \equiv [y_j^1, \ldots, y_j^N]$
- *macro-averaged (by example)*; N vectors of $\boldsymbol{z}_i \equiv \boldsymbol{y}_i$

test instances
$\boldsymbol{y}_1 = (1, 0, 1, 0)$ $\hat{\boldsymbol{y}}_1 = (1, 0, 0, 0)$
$\boldsymbol{y}_2 = (1, 0, 1, 1)$ $\hat{\boldsymbol{y}}_2 = (1, 0, 1, 1)$
$\boldsymbol{y}_3 = (1, 1, 1, 0)$ $\hat{\boldsymbol{y}}_3 = (1, 0, 0, 1)$

Measure	Value
EXACT-MATCH	$\frac{1}{3}1 = 0.333$
HAMMING-LOSS	$\frac{4}{12} = 0.333$
ACCURACY	$\frac{1}{3}(\frac{1}{2} + \frac{3}{3} + \frac{1}{4}) = 0.583$
F1-MICRO	$= 0.714$
F1-MACRO$^{\times L}$	$\frac{1}{3}(0.67 + 1.0 + 0.4) = 0.689$
F1-MACRO$^{\times N}$	$\frac{1}{4}(1.0 + 0.5 + 0.67 + 0.0) = 0.542$

Figure 3.1: An example of various multi-label evaluation measures.

which correspond to three separate measures, defined in Equations 3.4, 3.5, and 3.6.

$$\text{F1-MICRO}(D) = \text{F1}(\boldsymbol{z}, \hat{\boldsymbol{z}}) \tag{3.4}$$

$$\text{F1-MACRO}^{\times L}(D) = \frac{1}{L} \sum_{j=0}^{L} \text{F1}(\boldsymbol{z}_j, \hat{\boldsymbol{z}}_j) \tag{3.5}$$

$$\text{F1-MACRO}^{\times N}(D) = \frac{1}{N} \sum_{i=0}^{N} \text{F1}(\boldsymbol{z}_i, \hat{\boldsymbol{z}}_i) \tag{3.6}$$

The multi-label ACCURACY measure is effectively macro averaged by example. As pointed out by Tsoumakas et al. (2010), the micro averaged version of ACCURACY would be identical, however we note that they did not consider the macro average by label, which would be different.

In Figure 3.1 the common evaluation measures are illustrated with a toy problem of three test instances.

3.1.1 Label-based versus example-based evaluation

Different classifiers perform better under different evaluation measures. Chapter 4 and Chapter 5 will report on both methods which model label sets and methods which model individual labels. Intuitively, label-set based models will perform best under label-set based evaluation measures (like EXACT-MATCH, ACCURACY, and F1-MACRO$^{\times N}$) and label-based models will perform best under label-based evaluation measures (like HAMMING-LOSS and F1-MACRO$^{\times L}$), as also noticed in the literature (Dimou et al., 2009).

Because it is possible to select evaluation measures to benefit certain methods this thesis argues for multiple and contrasting evaluation measures in any multi-label experiment setup—as detailed in Section 3.4. In discussion of experiment results, later chapters will expand on, and make references to, the tendency of classifiers to be advantaged under different evaluation measures.

3.1.2 Evaluation without making predictions

Many multi-label methods initially predict a vector of real-valued confidence outputs for each test instance (e.g. posterior probabilities for each label), to which they apply a function to create a bipartition and thus a label-set prediction which can be measured under any of the measures reviewed above. Calibrating this function (usually a *threshold function*) is discussed in detail in Section 3.2. First, let us review an alternative: measuring confidence outputs directly.

Vens et al. (2008) detail the *area under the precision-recall curve* ($\text{AU}(\overline{\text{PRC}})$) for multi-label evaluation. Instead of calibrating a fixed threshold, this measure varies a threshold in steps $0.00, 0.02, \ldots, 1.00$ on the real-valued confidence outputs of a method and records micro-averaged precision and recall at each step, so as to produce a curve. The area under this curve is the $\text{AU}(\overline{\text{PRC}})$. They also consider the *average* area under the precision-recall curve, i.e. the curves for each individual label (a macro-averaged scheme), which may additionally be weighted by label frequency. They also make a strong case for precision-recall curves—as opposed to *ROC-curves*—for multi-label evaluation on account of the sparsity of labelling, i.e. the class-skew typical to multi-label data.

We notice that this technique is not restricted to the F-measure statistics and can be applied to any of the other measures reviewed so far in this chapter. Thus measures like the area under the *accuracy* curve are possible. To our knowledge this has not been mentioned in the literature. Although thresholds are still needed for predicting concrete label sets in real-world scenarios, additional curve-measures may prove interesting. We leave this investigation for future work.

3.1.3 Hierarchical and ranking evaluation

There exist several special measures for multi-label hierarchical evaluation, such as h-loss (Cesa-Bianchi et al., 2006), mainly to limit the effect on predictive performance caused by error propagation (see Section 2.5). These measures are obviously not relevant to a non-hierarchical context.

There are numerous measures for evaluating a ranking of label relevance;

for example *one-error*, *coverage*, and *ranking loss*; all reviewed in (Tsoumakas et al., 2010). These ranking measures consider the *order* of labels in a ranking. Although real-valued confidence outputs (that can be output by many multi-label methods) can be viewed as a ranking, we see this as only coincidental to achieving an accurate label-set prediction.

3.2 Threshold Functions in Multi-label Evaluation

Creating a bipartition of relevant and irrelevant labels is an ongoing theme in the multi-label literature. Threshold functions are relevant to all multi-label methods which can yield multi-label predictions from a vector of confidence outputs. Well-calibrated thresholds are more likely to result in balanced label-set prediction (where not too many, nor too few labels are predicted) and are therefore more more likely to be accurate.

3.2.1 Threshold functions

A multi-label prediction $\hat{\boldsymbol{y}}$ can be obtained from a confidence output vector $\hat{\boldsymbol{w}}$ with a threshold function f.

If threshold values are applied separately to the confidence output of each label, we call this *label-based thresholding*; i.e. $\hat{\boldsymbol{y}} = f_{t^L}(\hat{\boldsymbol{w}})$ (where t^L is an

L-vector of thresholds where each $t_j \in \mathbb{R}$) such that:

$$\hat{y}_j = \begin{cases} 1 & \text{if } \hat{w}_j \geq t_j \\ 0 & \text{if } \hat{w}_j < t_j \end{cases}$$

.

Such a label-based thresholding function has been used for some time in the context of text categorisation, e.g. (Yang, 2001), and was also reviewed by Fan and Lin (2007) in the context of general multi-label classification.

If a *single* threshold value is applied to all confidence outputs of all labels, we call this *label-set-based thresholding*; i.e. $\hat{\boldsymbol{y}} = f_t(\hat{\boldsymbol{w}})$ (where $t \in \mathbb{R}$ is a single value) such that:

$$\hat{y}_j = \begin{cases} 1 & \text{if } \hat{w}_j \geq t \\ 0 & \text{if } \hat{w}_j < t \end{cases}$$

We use the latter label-set based scheme. Considering multi-label data by label-set (i.e. by example) is arguably more natural than considering multi-label data by label, where each label is treated as if each label pertained to an independent binary problem. Furthermore, label-set based methods are arguably less prone to overfitting and only require the calibration of a single threshold value. Although, as reported by Fan and Lin (2007), label-based threshold functions can be optimised for label-based evaluation measures, under label-set based evaluation measures we found that label-*set*-based thresholding obtains similar predictive performance to label-based thresholding.

In many cases, it can be assumed that confidence outputs are posterior

probabilities, such that each $\hat{w}_j \in [0, 1]$. However, this is not always the case. Some ensemble methods, for example, vote on labels such that each $\hat{w}_j \in \mathbb{N}_0$. Comparing methods that output different kinds of confidence values complicates the task of setting an experiment-wide threshold. To take into account these cases, we normalise $\hat{\boldsymbol{w}}$, denoted as $\overline{\boldsymbol{w}}$, such that $\sum_{j=1}^{L} \overline{w}_j^i = 1.0$, which limits the effective range of potential thresholds (from 0.0 to 0.5) and therefore makes a threshold easier to set for use across all competing methods. We do note, however, that this procedure is undesirable when working with methods that only output posterior probabilities, since the semantics of such outputs are altered by such a transformation.

Consider the following two test instances classified under two different classification schemes A and B, giving different kinds of outputs, where column $f_t(\hat{\boldsymbol{w}}) \rightarrow \boldsymbol{y}$ gives the range of a threshold under which the correct label set ($\boldsymbol{y}$) is achieved:

	$\hat{\boldsymbol{w}} = h(\boldsymbol{x})$	$\overline{\boldsymbol{w}} = \text{NORM}(\hat{\boldsymbol{w}})$	$\boldsymbol{y}$	$f_t(\overline{\boldsymbol{w}}) \rightarrow \boldsymbol{y}$
$h_A(\boldsymbol{x}_1)$	$[0.94, 0.02, 0.00, 0.00]$	$[0.98, 0.00, 0.02, 0.00]$	$[1, 0, 0, 0]$	$0.02 \leq t < 0.98$
$h_B(\boldsymbol{x}_1)$	$[78, 0, 1, 0]$	$[0.99, 0.00, 0.01, 0.00]$	$[1, 0, 0, 0]$	$0.02 \leq t < 0.99$
$h_A(\boldsymbol{x}_2)$	$[0.90, 0.70, 0.00, 0.10]$	$[0.53, 0.41, 0.00, 0.06]$	$[1, 1, 0, 0]$	$0.06 \leq t < 0.41$
$h_B(\boldsymbol{x}_2)$	$[65, 80, 2, 9]$	$[0.42, 0.51, 0.01, 0.06]$	$[1, 1, 0, 0]$	$0.06 \leq t < 0.42$

Note that, after $\hat{\boldsymbol{w}}$ is normalised to $\overline{\boldsymbol{w}}$, any single threshold value of $0.06 \leq t < 0.41$ will lead to the correct classification in all four cases, despite the fact that the true prediction of both examples sets contain different numbers of labels and that the confidence outputs of methods A and B are of a totally different range and scale.

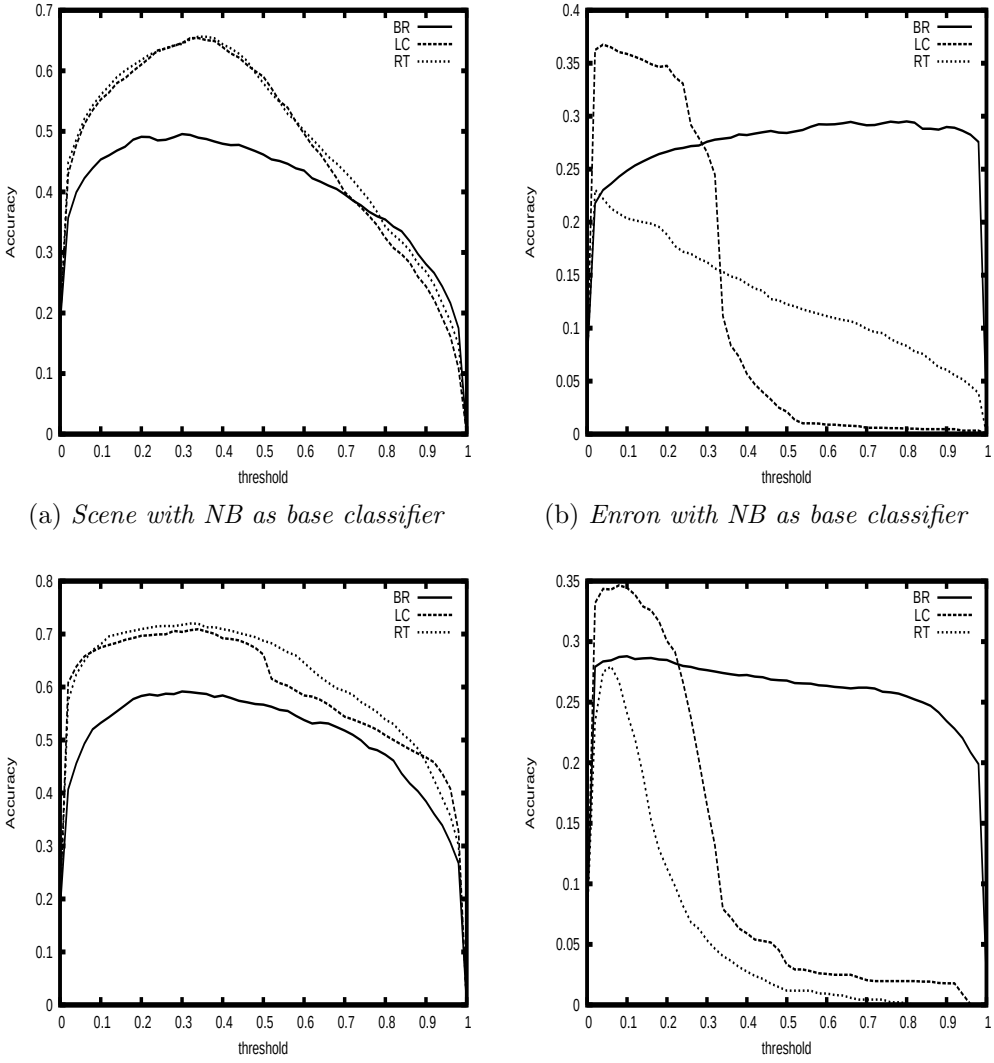
Empty-set predictions are possible (i.e. $|\hat{\boldsymbol{y}}| = 0$), particularly if thresholds

are calibrated too high. However, with a well-calibrated threshold, this is not generally an issue: we found that eliminating the empty-set condition by forcing a classification of at least one label (i.e. $|\hat{\boldsymbol{y}}| \geq 1$) using strategies detailed in (Spyromitros et al., 2008) made a negligible difference to our experiment results. Furthermore, in many cases it may be desirable to predict zero labels rather than force unconfident predictions.

3.2.2 Threshold calibration

Figure 3.2 displays the effect of a range of threshold values on the ACCURACY statistic of three fundamental multi-label problem transformation methods with both naive Bayes and support vector machines as base classifiers (note that logistic outputs are fit to the latter to provide smoother confidence outputs). Problem transformation methods will be reviewed in detail in Chapter 4, although these plots are not intended to compare either absolute or relative performance of the different classification methods, but rather to illustrate how threshold selection directly affects the predictive performance of different methods. The figure shows how performance under a particular threshold is related to the dataset, the multi-label method, and the single-label base classifier employed by that method. Note that outputs were not normalised to 1.0 is this case, to show that these effects occur independently of this optional procedure. The need for threshold calibration on a per-dataset and per-method basis is clear.

In multi-label circles (for example in (Tsoumakas and Vlahavas, 2007)), it is common to consider an *arbitrary threshold*, e.g. 0.5. Yang (2001) discussed some more advanced methods of threshold calibration for threshold func-



(a) *Scene with NB as base classifier* (b) *Enron with NB as base classifier*

(c) *Scene with SVMs as base classifier* (d) *Enron with SVMs as base classifier*

Figure 3.2: ACCURACY with respect to threshold for methods `BR`, `LC`, `RT` (see Chapter 4) on *Scene* and *Enron* with naive Bayes (NB) and support vector machines (SVMs) as base classifiers.

tions used in text categorisation circles in a binary context i.e. label-based threshold functions:

- An *internal validation* method (IVAL), where each threshold $t_j \in t^L$ is calibrated like any parameter, by testing a set of values on a selection or selections of the training data, and selecting the value (or average of the values) which results in the best performance under some chosen measure of evaluation.

- The *proportional cut* method (PCUT), where each threshold t_j is chosen according to the frequency at which label y_j is relevant in the training data (to approximate the same frequency in the testing data).

These methods can also be implemented in a label-set-based fashion (which is our preference, as explained in Section 3.2.1).

It is clear from Figure 3.2 that arbitrary thresholds are inappropriate in a general setting. IVAL can produce good thresholds but at a potentially huge computational cost, as was our experience in (Read, 2008). In all the large-scale experiments of this thesis (in Chapters 6 and 7) we used a label-set-based PCUT method, which we found calibrates thresholds as effectively as IVAL but at a negligible computational cost. This is despite the fact that, unlike IVAL, PCUT is not tailored to any particular evaluation measure. Thus, PCUT is a fast and effective method suitable for general use in experimental evaluations.

Given threshold function $f_t : \mathbb{R}^L \rightarrow \{0,1\}^L$ under a threshold t, and each $\overline{\boldsymbol{w}}_i$ obtained beforehand (for computational efficiency) for each test instance $\boldsymbol{x}_i | i = 1, \ldots, G$, label-set-based PCUT chooses the t which minimises the

difference in label cardinality between the training data and the classified test data:

$$\text{PCUT} = \underset{t}{\operatorname{argmin}} \left\| \text{LCARD}(\mathcal{D}) - \left(\frac{1}{G} \sum_{i=1}^{G} |f_t(\overline{\boldsymbol{w}}_i)| \right) \right\|$$

3.2.3 Alternatives to threshold functions

Threshold functions are widely used in the literature, but some methods may only provide label rankings or poor quality confidence outputs (which are difficult to calibrate a threshold on). There are other ways to acquire a concrete label-set prediction without a threshold; essentially by predicting the number of labels for each test instance (i.e. the top $|\hat{\boldsymbol{y}}_i|$ labels from a ranking for each test instance $\boldsymbol{x}_i$). Existing methods include:

- Assuming prior knowledge of each $|\boldsymbol{y}_i|$ (Luo and Zincir-Heywood, 2005)
- The *k-per-doc* method: each test example is assigned exactly k labels ($\forall \hat{\boldsymbol{y}}_i : |\hat{\boldsymbol{y}}_i| = k$) (Lewis, 1992; Yang, 2001)
- A meta method like *regression* or a multi-class classifier (Tang et al., 2009)
- *Calibrated Label Ranking* (CLR): partitioning a ranking with a virtual label (Fürnkranz et al., 2008)

Assuming prior knowledge of the number of labels for each test instance is not a valid assumption in the real world, and the k-per-doc method is likely to perform well only where the number of labels per example is extremely regular throughout the data (our studies in Chapter 2 indicate the rarity of

such data). Tang et al. (2009) use a meta multi-class classifier with classes $\{2, \ldots, L\}$ to predict the number of labels for each $\hat{\boldsymbol{y}}_i$. Fürnkranz et al. (2008) introduce CLR (which is, arguably, also a meta method). CLR draws a label set prediction from a ranking by using a virtual label. This label is calibrated in the ranking so that the relevant labels lie before it. Calibration is done with $L + 1$ binary classifiers. Both these methods scale with L; any meta method implies an extra computational cost.

3.3 Log Loss

In the previous section we saw how a threshold function can produce label-set predictions from confidence outputs, which can then be evaluated by a variety of measures. It is also possible to evaluate the confidence outputs directly. This section details our *log loss* measure for multi-label evaluation, which does exactly that.

Even when all competing methods in an experimental evaluation use the same threshold function, there are limitations: variations in this function can produce different results under different methods, by allowing more or fewer labels to be relevant to test instances. To ensure that methods are not unfairly advantaged by a particular thresholding scheme, we argue that a large-scale multi-label evaluation should include non-threshold dependent measures.

Ranking measures do not require a threshold, but are not suitable, since we are not interested in the order of label relevance. Boutell et al. (2004) develop several parameterised versions of ACCURACY to allow different costs

for false positives and true negatives. This does not require a threshold, but instead requires additional parameters. In (Read et al., 2009b) we introduced *log loss* to multi-label evaluation, which is a less complicated and more general measure, which supplements AU($\overline{\text{PRC}}$) as a measure not requiring a threshold.

Under log loss, each label error is graded by the confidence at which it was predicted: predicting false positives with low confidence induces logarithmically less penalty than predicting with high confidence. Log loss can be defined as in Equation 3.7.

$$\begin{gathered}\text{LOG-LOSS}(\mathcal{D}) = \frac{1}{NL}\sum_{i=1}^{N}\sum_{j=1}^{L}\min(-\text{LOG-LOSS}(\hat{w}_j^i, y_j^i), \ln(N)) \\ \text{where LOG-LOSS}(\hat{w}, y) = \ln(\hat{w})y + \ln(1-\hat{w})(1-y)\end{gathered} \tag{3.7}$$

This measure includes a dataset-dependent maximum of $\ln(N)$ to limit the magnitudes of the penalty (where N is the number of test examples). Such a limit has been used in other domains where a log loss function is used, for example (Tan and Dowe, 2003; Schapire and Singer, 1999), and serves to smooth the values to prevent a small subset of poorly predicted labels from greatly distorting the overall error. LOG-LOSS assumes that each confidence value is between 0.0 and 1.0 inclusive, i.e. essentially posterior probabilities such that $\forall w_j : w_j \in [0, 1]$ (note that is a different normalisation process to the one outlined above for thresholding purposes). As a loss metric, the best possible score for LOG-LOSS is 0.0.

LOG-LOSS is distinct from other measures because it punishes poor prediction confidences directly, and with respect to the degree of confidence (punishing worse errors more harshly). It is thus immune to the effects of

Where $\boldsymbol{y} = [1, 0, 0, 1], \hat{\boldsymbol{w}} = h(\boldsymbol{x}) = [0.5, 0.0, 0.4, 0.2]$:

$\hat{\boldsymbol{y}} = f_t(\hat{\boldsymbol{w}})$	t	ACCURACY	HAMMING-LOSS	LOG-LOSS
$[1, 0, 0, 0]$	0.45	0.50	0.25	3.22
$[1, 0, 1, 0]$	0.35	0.33	0.50	3.22
$[1, 0, 1, 1]$	0.15	0.67	0.25	3.22

Figure 3.3: Illustration of ACCURACY, HAMMING-LOSS, and LOG-LOSS for one example $(\boldsymbol{x}, \boldsymbol{y})$ under different thresholds (t), given a set of prediction confidences $\hat{\boldsymbol{w}}$. Note that we assume that $N = 100$ (so that the dataset-dependent limit of $\ln(N)$ may be realistic).

threshold calibration and contrasts well with evaluation measures which may reward guessing labels under low confidence, or ignoring labels even under high confidence. Also, because LOG-LOSS punishes worse errors much more harshly, it can provide larger margins of contrast between competing methods. Essentially, LOG-LOSS is a kind of information-gain measure which tells us about how close confidence predictions are to the real label values.

LOG-LOSS is illustrated in Figure 3.3 alongside other measures on a toy problem of a single test instance. This figure illustrates how a low threshold under ACCURACY (and thus guessing two extra labels, even though one of them is a false positive) can be beneficial and, likewise, a high threshold under HAMMING-LOSS (where predicting one label is better than predicting two in this case). Under LOG-LOSS the classifier has been penalised for its relatively high confidence on the third label (which is *not* relevant to the instance), and its low confidence on the fourth (which *is* relevant).

By encouraging quality prediction confidences, LOG-LOSS can be useful for related tasks where the *degree* of label relevances is important: for a news article about `science` and `sports` with a central theme of `science`,

we would expect a higher relevance for the label `science`. Likewise, an image may contain both `beach` and `mountains`, but pertain to more of one than the other. Unlike ranking, label relevancy is gauged by real values. We leave further exploration of this alternative for future work.

Like any measure in multi-label evaluation, LOG-LOSS should not stand alone, but rather be used along with other contrasting measures to form part of a balanced multi-label evaluation. The following section elaborates the experiment setup for the major evaluations of this thesis.

3.4 Experimental Setup

Like in the traditional single-label context, extensive experimental evaluation is necessary to empirically confirm method performance. It is worth discussing experimental setup specifically with respect to multi-label contexts, since this area is comparatively new and underdeveloped.

As a result of its relative novelty multi-label classification does not yet have a standard collection of datasets and many evaluations in the literature use only one, or a very limited collection. Another issue has been the lack of comparison between methods: only recently have authors begun to compare their work to a variety of other competitive methods from the literature.

Due to the extra label dimension of multi-label data any multi-label experimental setup should compare method performance with multiple and contrasting evaluation measures. The major evaluations of this thesis use the following measures:

Measure name	Measure type	Requires threshold
ACCURACY	label-set based	yes
EXACT-MATCH	label-set based	yes
AU($\overline{\text{PRC}}$)	label based	no
F1-MACRO$^{\times L}$	label based	yes
LOG-LOSS	label based	no

Additionally, we consider running times. All thresholds are calibrated consistently and universally using the label-set based proportional cut (PCUT) method detailed in Section 3.2. Note that sometimes, for small pilot experiments, we display the results of fewer measures for brevity.

During this research we developed our own multi-label framework to implement and evaluate our methods as well as other methods from the literature. This framework, which includes the classification and multi-label methods detailed in this thesis, is a *m*ultilabel *e*xtention of WEKA (Hall et al., 2009) (MEKA) is available at: `http://sourceforge.net/projects/meka/`.

In recent years, other software has been developed, most notably the MULAN framework[1] (Tsoumakas et al., 2009b)—also based on WEKA—which implements well-known methods from the literature. MEKA has a wrapper for MULAN so as to include these methods in our evaluations.

Our standard setup is 5×2 fold cross validation (five rounds; two folds) on standard-sized datasets, and a 60%/40% train/test split on the larger datasets (due to the large running times incurred by the slower methods). The approximate partition between 'standard' and 'large' datasets was made

[1] `http://sourceforge.net/projects/mulan/`

in Table 2.1 with large datasets following the horizontal line.

We use two statistical comparisons. An appropriate test for comparing the performance of multiple methods directly against a particular method is the *corrected paired t-test* (Nadeau and Bengio, 2003); significance is determined under a p value of 0.05. Additionally, we used the *Nemenyi test* (Demšar, 2006) to compare between multiple classifiers. The Nemenyi test has been observed as not as sensitive (Cheng and Hüllermeier, 2009) but, since it tests significance over a collection of datasets, it has the additional advantage that it can be employed on the results of a train/test split evaluation. We also consider methods' average ranks.

We run our experiments on 3 GHz machines allowing for 2 Gigabytes of memory. We define *did not finish* (DNF) unable to complete on a given dataset, either due to lack of memory or exceeding the time of ten days.

Minor deviations from this setup and other experiment details algorithm specifics, parameter configurations, and so on will be made explicit in the main experimental sections.

In the course of this research we have several times expanded the collection of publicly available datasets, compared to notable methods in the literature, and used multiple evaluation measures. We continue to do so in the experimental sections of this thesis. Two major empirical evaluations appear in Chapters 6 and 7.

Chapter 4
Problem Transformation

Problem transformation is the process whereby a multi-label problem is transformed into one or more single-label problems. In this scheme, single-label classifiers are employed, and their single-label predictions are transformed into a multi-label prediction. An alternative to problem transformation is *algorithm adaptation* where an existing single-label algorithm is adapted directly for the purpose of multi-label classification. These two approaches are discussed in (Tsoumakas and Katakis, 2007).

The prime advantage of problem transformation is flexibility. By abstracting away from a specific classifier, any off-the-shelf single-label classifier can be used to suit requirements. Depending on the problem context, some classifiers may demonstrate better performance than others.

Algorithm adaptation methods are usually designed with a specific domain in mind. For example, decision trees are typically used on biological datasets (Clare and King, 2001; Vens et al., 2008) where they perform well and provide interpretable models, while Bayes-based mixture models are commonly used specifically on text data (McCallum, 1999). Additional work on algorithm adaptation is reviewed in Chapter 5. The advantage of problem

transformation is that it can abstract away from classifier specifics and be more generally applicable by focussing on issues relevant to all multi-label domains such as modelling label correlations.

Moreover, algorithm adaptation methods almost invariably involve problem transformation *internally*, which can often be applied generically to a range of classifiers. Therefore, many algorithm adaptation methods can also be applied generally to *external* problem transformation approaches, which are the focus of this thesis.

This chapter reviews, compares, and contrasts fundamental problem transformation methods that are found in variations and combinations throughout the multi-label literature. Chapter 5 will provide a general review of approaches in the literature, many of which use directly—or draw inspiration from—these fundamental methods.

4.1 The Problem Reduction Method

Before we review the fundamental methods, let us quickly look at a primitive form of problem transformation: *problem reduction*, reviewed in (Tsoumakas and Katakis, 2007). By eliminating certain labels, or examples, it is possible to reduce a multi-label problem to a single-label problem. Essentially there are two basic problem-reduction approaches, either

1. remove all examples $(\boldsymbol{x}, \boldsymbol{y})$ where $|\boldsymbol{y}| > 1$ from the dataset; or
2. remove $(|\boldsymbol{y}| - 1)$ labels from each label set $\boldsymbol{y}$.

Both approaches result in a dataset with strictly one label to each example $(\forall(\boldsymbol{x}, \boldsymbol{y}) \in D : |\boldsymbol{y}| = 1)$, i.e. a single-label problem.

It is clear, however, that in both cases considerable information may be lost, and performance will suffer accordingly. For this reason, such primitive methods have never been seriously considered in the literature, and can be safely discarded from further analysis.

We now proceed to the fundamental problem transformation methods which are widely used throughout the literature. The advantages and disadvantages of each method will be further discussed in Chapter 5.

4.2 Fundamental Problem Transformation

4.2.1 The Binary Relevance method (BR)

The most well-known and widely documented problem transformation method is the *binary relevance* method (BR) (Tsoumakas and Katakis, 2007; Godbole and Sarawagi, 2004; Zhang and Zhou, 2005). BR transforms any multi-label problem into L binary problems[1]. Each binary classifier is then responsible for predicting the association of a single label (one binary problem for each label). Using the notation from Section 1.4, a formal overview of the process is as follows.

Transformation $(\boldsymbol{x}, \boldsymbol{y}) \rightarrow \{(\boldsymbol{x}, y_j) | j = 1, \ldots, L\}$

Classifier $\boldsymbol{h} = (h_1, \ldots, h_L) : \mathcal{X} \rightarrow \{0, 1\}^L$

Classification $\hat{\boldsymbol{y}} = [y_1, \ldots, y_L] = [h_1(\boldsymbol{x}), \ldots, h_L(\boldsymbol{x})]$

[1] BR-type methods are occasionally referred to in the literature as ensemble methods because they involve multiple binary classifiers. However, none of these classifiers is multi-label capable. Throughout this thesis we use the term *ensemble* strictly in the sense of an *ensemble of multi-label methods*.

Thus, BR is made up of L classifiers $h_j|j = 1, \ldots, L$. Each h_j is trained with all $(\boldsymbol{x}_i, y_j^i)|i = 1, \ldots, N$. For each test instance, each h_j predicts the binary association of the jth label.

Although conceptually simple and relatively fast, it is widely recognised that BR does not explicitly model label correlations (Park and Fürnkranz, 2008; Elisseeff and Weston, 2001; Yan et al., 2007; Godbole and Sarawagi, 2004; Zhang and Zhou, 2007b) since it constructs a decision boundary individually for each label. BR can also be affected by *class-imbalance* (Ráez et al., 2004) since, due to the typical sparsity of labels in multi-label data, each binary classifier is likely to have far more negative examples than positive.

4.2.2 The Pairwise Classification method (PW)

Whereas BR is a "one-vs-rest" paradigm (as it is known in multi-class classification) where one classifier is associated with the relevance of each label, *pairwise* classification (PW) is a "one-vs-one" paradigm where one classifier is associated with each *pair* of labels. Hence, instead of L binary problems, $P = \frac{L(L-1)}{2}$ binary problems are formed: one for each pair. Typically (Hüllermeier et al., 2008; Fürnkranz et al., 2008), each pairwise problem is made up of examples with which either labels (but not both) are associated, thus forming a decision boundary for these two labels. The process contains the following elements.

Transformation $(\boldsymbol{x}, \boldsymbol{y}) \rightarrow \{(\boldsymbol{x}, y_j)|y_j \otimes y_k = 1, 1 \leq j < k \leq L\}$

Classifier $\boldsymbol{h} : \mathcal{X} \rightarrow \{0, 1\}^P$

Classification $\hat{\boldsymbol{y}} = f(h_{j,k}(\boldsymbol{x})|1 \leq j < k \leq L)$

where $f : \{0,1\}^P \rightarrow \{0,1\}^L$ is a function which, given an input instance, takes the output of all pairwise classifiers as input and outputs a multi-label prediction for that instance.

Note that, unlike the relevance classifications of BR, the pairwise comparisons of PW do not result immediately in a label set, but rather a set of *pairwise preferences*, and this additional function f is necessary to create a multi-label prediction (also discussed in Chapter 3). In (Park and Fürnkranz, 2008) the votes of all pairwise preferences are aggregated to produce confidence outputs, to which a threshold function can be applied. However, threshold functions are surprisingly uncommon for PW methods, perhaps because associated confidence outputs are difficult to calibrate a threshold for, as suggested by Petrovskiy (2006). The widely cited CLR method (Fürnkranz et al., 2008) calibrates a virtual label to bipartition a ranking for producing label-set predictions with a PW method.

Time complexity is an issue for PW: it is quadratic with respect to the number of labels. PW approaches have also been criticised for not dealing well with overlapping labels and struggling to establish disjoint assignments in the multi-label context (Petrovskiy, 2006; Ráez et al., 2004).

4.2.3 The Label Combination method (LC)

Another fundamental problem transformation method is the *label combination* method (LC) or *label power-set* method. LC treats all label sets as atomic (single) labels to form a single-label problem in which the set of single labels represents all distinct label sets in the multi-label training data. Thus the

label of each single-label example is in fact a multi-label set, and hence the following notation.

Transformation $(\boldsymbol{x}, \boldsymbol{y}) \rightarrow (\boldsymbol{x}, c) | c \equiv \boldsymbol{y}$

Classifier $h : \mathcal{X} \rightarrow \mathcal{Y}'$ where $\mathcal{Y}' = \{c | c \equiv \boldsymbol{y}, \exists \boldsymbol{x} : (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}\}$

Classification $\hat{\boldsymbol{y}} = \hat{c} = h(\boldsymbol{x})$

Although LC can take into account label correlations directly, unlike the binary models of BR and PW, it can only classify new examples with label sets it has already seen in the training set i.e. it can overfit the training data. LC can also be computationally complex (Tsoumakas and Vlahavas, 2007; Read et al., 2008; Cheng and Hüllermeier, 2009; Veloso et al., 2007) since it requires as many class labels in the single-label transformation as there are distinct label sets in the training data—worst case $\text{MIN}(L, 2^L - 1)$—, and many of these labels are likely to be very sparse with respect to training examples.

4.2.4 The Ranking and Threshold method (RT)

The *ranking and threshold* method (RT) trains a multi-class problem, and relies on a thresholding function to make multiple predictions per instance. We described thresholding in Section 3.2: all labels associated with a confidence value greater than a threshold form the multi-label prediction. RT is detailed in (Tsoumakas and Katakis, 2007): each multi-labelled example is decomposed into multiple single-label (multi-class) examples—one for each relevant label—by duplicating the instance space of each training example

and assigning one of the relevant labels to each new example. The resulting L-class dataset can be used to train any single-label multi-class-capable classifier. The process is as follows.

Transformation $(\boldsymbol{x}, \boldsymbol{y}) \rightarrow \{(\boldsymbol{x}, c) | c \equiv j, y_j = 1\}$

Classifier $h : \mathcal{X} \rightarrow \mathbb{R}^L$

Classification $\hat{\boldsymbol{y}} = f_t(h(\boldsymbol{x}))$

where $f_t : \mathbb{R}^L \rightarrow \{0, 1\}^L$ is a threshold function under threshold t.

Although threshold functions are widespread, `RT`'s transformation is rarely used in the literature. A disadvantage of this classification method is the reliance on classifier confidence outputs, which may be difficult to threshold, or may not be output at all by some classifiers. Also—duplicate instances being associated with different class labels in the transformed data complicate modelling decision boundaries. Similarly to `BR`, this method does not explicitly model label correlations.

4.3 Problem Transformation Complexity

Table 4.1 provides a summary of the fundamental problem transformation methods and the dataset dimension with the strongest influence on their complexity. The methods clearly vary in terms of their worst case scenarios, memory use, and running time.

We see that `BR` scales linearly with L, whereas `PW` scales quadratically, although each of `PW`'s binary classifiers is likely to be trained on fewer than N examples. `LC` only instantiates one single-label classifier, but the number

Table 4.1: The problem transformation methods, the dataset measurement to which they predominantly scale, the number of single-label classifiers employed, and the number of single-labels and the number of examples each of those classifiers deals with.

method	scales with	#classifiers	#lbls/classifier	#exs/classifier
BR	L	L	2	N
PW	L	$L(L-1)/2$	2	$\leq N$
LC	$\#\textsc{Dist}(\mathcal{D})$	1	$\#\textsc{Dist}(\mathcal{D})$	N
RT	$\textsc{LCard}(\mathcal{D})$	1	L	$N \times \textsc{LCard}(\mathcal{D})$

where $\#\textsc{Dist}(\mathcal{D}) = \left|\{\boldsymbol{y} | \exists \boldsymbol{x} : (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}\}\right|$

of class labels of this classifier is bound to the number of distinct label sets in the training data (#Dist), which may be as high as N (the number of distinct *examples* in the training data). RT also involves only a single classifier and the number of class labels grows only linearly to the number of labels, but in this case the number of training examples grows with the LCard of the training set.

4.4 Single-label Base Classifier Selection

Problem transformation methods offer the flexibility of using any single-label base classifier. It is important, therefore, to consider the effect that the choice of single-label classifiers has, both on predictive performance and the computational complexity of the overall process.

Table 4.2 provides a list of well-known single-label classifiers which have been used in the multi-label literature, either employed by problem transformation methods or in modified form as algorithm adaptation methods (as Chapter 5 will review).

Table 4.2: Single-label classifiers used in the multi-label literature.

Key	Name [Citation]	(WEKA implementation)
NB	Naive Bayes (John and Langley, 1995)	NaiveBayes
SVM	Support Vector Machines (Platt, 1999)	SMO
DT	Decision Trees (Quinlan, 1986)	J48
kNN	k Nearest Neighbor (Wang et al., 2000)	IBk
NN	Neural Networks (Haykin, 1998)	MultilabelPerceptron
DR	Decision Rules (Cohen, 1995)	JRip

SVMs are a popular choice (Tsoumakas and Vlahavas, 2007; Tang et al., 2009; Godbole and Sarawagi, 2004; Read et al., 2008) due to their reputed predictive power, although for multi-class problems (such as those produced by LC and RT) SMO uses an internal pairwise-coupling method which scales exponentially with respect to the number of class labels. k-nearest neighbor (kNN) methods are also popular in multi-label classification (Zhang and Zhou, 2007a; Spyromitros et al., 2008; Cheng and Hüllermeier, 2009) but, since every kNN query scales linearly in the number of training examples, this can be an expensive scheme for large training sets. Naive Bayes has been used where other methods would be too computationally expensive (Tsoumakas et al., 2008).

To the best of our knowledge, only one general study of single-label algorithms in the context of problem transformation has been carried out (by Tsoumakas and Katakis (2007)). We extended this study both with respect to the number of problem transformation methods and the range of base classifiers. We tested each of the four main problem transformation methods (described in this chapter) with each of the six single-label base classifiers (listed in Table 4.2)—giving a total of 24 classification schemes—in a

train/test scenario on the first 10 datasets of Table 2.1. Table 4.3 displays some results—a list of the methods which got the top rank on at least one dataset for at least one of the two evaluation measures.

SMO is found in the list of the top methods under all problem transformation methods except RT. RT performs well with, for example, NaiveBayes, which provide smoother confidence predictions to which a threshold is more effectively applied. J48 appears on the list under binary transformations (BR and PW). Only partially observable in the table is that MultilayerPerceptron and JRip only perform well and under time and memory constraints under BR (and even then, not always).

SMO provides the best trade off between predictive performance and computational complexity: it is slower than some other classifiers, but fast enough to finish within limits even under LC; it gets by far the most top-rankings across datasets. For that reason it is our classifier of choice for most of the experimental evaluation in this thesis. Alternative choices would be decision trees or naive Bayes where, in terms of predictive performance, the former (J48) is most beneficial on binary transformations and the former (NaiveBayes) is most beneficial on multi-class transformations.

Not shown in the table (for brevity) is that different methods perform distinctively under different datasets, which is further justification for working with problem transformation methods in a general setting rather than being confined to specific algorithms that may be suitable only for specific domains.

Table 4.3: The best 12 combinations of problem transformation (PT) and single-label base-classifier (SL); in terms of the number of wins across 10 datasets (note ties are possible) by ACCURACY and F1-MACRO$^{\times L}$, the number of datasets each method could finish on within 24 hours, and the number of wins as proportions of the datasets that were finished.

		no. of wins (/10)			% of wins	
PT	SL	by ACC.	by F1$^{\times L}$	#Fin.	by ACC.	by F1$^{\times L}$
LC	SMO	5.0	3.0	10	0.50	0.30
BR	Perceptron	1.0	1.0	3	0.33	0.33
PW	SMO	1.0	1.0	4	0.25	0.25
BR	JRip	1.0	2.0	10	0.10	0.20
PW	J48	1.0	0.0	10	0.10	0.00
LC	NaiveBayes	1.0	0.0	10	0.10	0.00
BR	J48	0.0	1.0	9	0.00	0.11
RT	NaiveBayes	0.0	1.0	10	0.00	0.10
BR	SMO	0.0	1.0	10	0.00	0.10
LC	IBk	0.0	1.0	10	0.00	0.10
BR	IBk	0.0	1.0	10	0.00	0.10
RT	IBk	0.0	1.0	10	0.00	0.10

4.4.1 Single-label base classifier optimisation

The performance of many single-label classifiers can often be improved by optimising their respective parameters. For example, Dimou et al. (2009) employ the `LIBSVM` library to tune the parameters of SVMs. However, fine-tuning parameters is intensive, classifier-specific, and not directly related to multi-label classification performance, and we instead use default parameters for the single-label classifiers wherever possible (such as those provided by WEKA (Hall et al., 2009)).

Chapter 5
Prior Work

Chapter 4 reviewed problem transformation approaches for multi-label classification. As well as standalone methods, these approaches constitute the core of many algorithm adaptation methods. This chapter reviews the most relevant and most significant contributions to the multi-label literature.

The first part of this chapter reviews work embracing the most common problem transformation paradigms. Following sections review algorithm adaptation methods. The final part summarises prior work specifically with respect to efficiency and scalability to large datasets.

5.1 BR Methods

BR is arguably the most well-known multi-label method. It is simple to implement and has been used as a baseline method throughout the multi-label literature. In Chapter 4, we saw that the fact that BR does not explicitly model label relationships was widely claimed, and that this is often used to justify alternative approaches. Some methods, however, combat this problem of BR directly.

In the main contribution of (Godbole and Sarawagi, 2004), BR classification outputs are stacked into a separate meta classifier along with the original feature space, creating a two-stage classification process which can learn label correlations in the data. We refer to this method as *meta*-BR (MBR). The meta process implies an extra iteration on both training and test data, as well as internal classifications on the training data to acquire the label outputs for the secondary training step. Tsoumakas et al. (2009a) exploit the redundancy among the models of the MBR scheme to improve the overall efficiency of the process.

A method reviewed in (Schapire and Singer, 1999) uses a meta scheme with a Hamming distance metric to map the confidence predictions of a single-label classifier to label sets which have been observed in the training data. The subset with the shortest Hamming distance to the predictions is chosen as the predicted set. When applied to the binary outputs of BR we refer to this method as *subset-mapped* BR (SMBR).

The *classifier chains* (CC) problem transformation method, which we present in Chapter 7, models label correlations without leaving the BR paradigm. It uses a stacking-like process, but unlike MBR, this process runs in a chain and therefore does not require a meta step or internal cross validation for training.

A BR-based boosting algorithm is introduced in (Yan et al., 2007). Binary models are trained on subsets of the example and attribute spaces (N and M). It reduces redundancy in the learning space and by sharing binary models between labels and thereby also reduces complexity. This is a related aim and approach to the work in (Tsoumakas et al., 2009a), but involving a

boosting ensemble instead of MBR's stacking method.

The work in (Ji et al., 2008) is a general framework for extracting shared subspaces in a BR approach, where a second part to the method depends on the representations in this shared subspace. This shared subspace models label correlations. However, despite using an approximation algorithm, the resulting method is computationally expensive, which is reflected in the experimental setup, where only relatively small samples of 1000 training instances are used.

Apart from not modelling label correlations, a secondary issue relating to BR is *class-label imbalance*. Ráez et al. (2004) focus on this issue with respect to text categorisation by overweighting positive examples in the BR models. This work is also involved in a real-time environment and on large collections, and the authors discover that classification speed can be improved with marginal effect on predictive performance by ignoring rare class labels altogether (at least with respect to their text data corpus).

5.2 PW Methods

PW-methods have predominantly been used in ranking schemes (as mentioned in Section 4.2.2) since, unlike BR, the classification phase results in a set of pairwise preferences which give rise more naturally to a ranking than a label set prediction. A well known example of PW-ranking is the *ranking by pairwise comparison* scheme (RPC) (Hüllermeier et al., 2008), which obtains a ranking by counting the votes received by each label.

The CLR scheme (Fürnkranz et al., 2008) arguably best completes the task

of multi-label classification by extending RPC with *calibrated label ranking* to create a bipartition of relevant and irrelevant labels (as mentioned in Section 3.2.3): a virtual label partitions a ranking into relevant and irrelevant labels to form a concrete label-set prediction for any test instance. An additional L binary classifiers approximating the standard BR problem are required to calibrate this label at training time. CLR has become a heavily cited work, and we compare with it in our experimental evaluations.

Due to the inherent large number of classifiers under a PW scheme (quadratic with respect to L), most PW approaches use efficient single-label base classifiers to improve scalability. CLR, for example, was presented with simple perceptrons. MLPP (Mencía and Fürnkranz, 2008) is another PW scheme that trains one perceptron for each possible class-label pair. As explained by the authors of this work—although MLPP performs better than the related BR-based perceptron algorithm MMP (see Section 5.7)—as a PW method, it scales quadratically with L rather than linearly.

It was shown specifically by Loza Mencía and Fürnkranz (2008), that a PW-based classifier can scale to large L by using simple perceptrons. This classifier is DMLPP, a modified version of MLPP which includes a special adaptation: rather than having to maintain the $(L(L-1))/2$ models normally associated with PW, it instead keeps all examples in memory and builds models dynamically (i.e. a lazy scheme) for each prediction it is required to make. Hence, time and memory in terms of L is sharply reduced in a trade for increased time complexity in terms of N at prediction time.

These latter two schemes focussed specifically on obtaining and evaluating a multi-label ranking (a CLR scheme could have been used—but at a

computational cost), and were applied specifically to text-categorisation. PW loses some of the advantages of an external problem transformation method if it is restricted to certain classification schemes. DMLPP could be viewed as an algorithm-adaptation approach and, as such, is inevitably restricted to certain base classifiers and domains.

PW has been criticised for failing to establish disjoint label assignments in the multi-label context, as noticed by Petrovskiy (2006), who deals better with overlapping-labels in PW by accompanying each binary model by two additional probabilistic models to isolate the overlapping attribute space. A threshold function is then applied to the resulting pairwise probabilities to produce a concrete label-set prediction. This approach shows high predictive performance in comparison with some other methods, although the authors discuss a computational bottleneck of this method on large datasets.

An RPC-based PW approach introduced in (Park and Fürnkranz, 2008) is able to learn from label constraints, such as that the presence of a label always implies the presence of another label, or that two labels never occur together. Such constraints are found most commonly in hierarchical and structured data.

5.3 LC Methods

LC was used in the relatively early work by Boutell et al. (2004) on image classification with the *Scene* dataset. *Scene*, however, is a relatively small dataset. The fact that LC scales quite poorly (quadratically with the number of distinct label sets) has inhibited its widespread use as an off-the-shelf

method.

RAkEL (RAndom K-labEL subsets) (Tsoumakas and Vlahavas, 2007) trains m random subsets of k labels using LC models under an ensemble scheme using a threshold to produce label-set predictions from the votes of all m models at testing time. RAkEL's complexity is thus limited by k instead of L, where $k \leq L$. This method has become one of the most well known in the multi-label literature and a standard benchmark in many evaluations, including our own.

In Chapter 6 we present the *pruned sets* method (PS) (Read, 2008; Read et al., 2008), which is also based on LC. PS reduces the complexity of LC by pruning away infrequent label sets and thereby, under the typical label-skew of multi-label data, greatly reducing the number of sets which must be represented as class labels. Our ensemble version of this method (EPS) shows even higher predictive performance while still maintaining scalability.

5.4 Decision Trees and Boosting

Decision trees have been modified to support multi-label classification, and are especially popular in bioinformatics on account of their interpretability (Struyf et al., 2005). In the most well known decision-tree adaptation—(Clare and King, 2001)—the authors modify the expression for entropy of the C4.5 algorithm to allow the tree to predict a label vector, as opposed to a single class label. In this way, a tree model can give a multi-label prediction at the leaves. This adaptation was applied to genomics data. Decision tree methods are also frequently used in hierarchical settings, such as by Struyf

et al. (2005) and Vens et al. (2008).

The BoosTexter system (Schapire and Singer, 2000) was for some time considered the state-of-the-art in multi-label classification and ranking. BoosTexter supplies `AdaBoost.MH` and `AdaBoost.MR`, both of which are a multi-label adaptation of the well-known `AdaBoost` boosting paradigm. An initial transformation phase transforms each example $(\boldsymbol{x}, \boldsymbol{y})$ into L examples $(\boldsymbol{x}, c_1), \ldots, (\boldsymbol{x}, c_L)$ where $c_j = 2y_j - 1$ (i.e. $c_j \in \{-1, +1\}$). Over this distribution, the weight of the incorrectly classified training examples are incremented as per regular boosting. `AdaBoost.MH` minimizes Hamming loss (where any positive signs (+1) become the label-set prediction) and `AdaBoost.MR` minimises the ranking loss (i.e. a ranking focussed method). Improving these algorithms, including threshold selection, was a focus of the work in (Kiritchenko, 2005). An extended version—adapted decision tree boosting (*ADT*`Boost.MH`)—was presented in (De Comité et al., 2003).

These `AdaBoost` methods have mainly been used in bioinformatics applications (where boosting and decision trees are particularly popular (Kiritchenko, 2005)). Although they can also work with textual datasets, they scale poorly with L and can fail to perform well on sparse data (Freund and Schapire, 1999). This family of methods has been recognised as having high computational complexity (Petrovskiy, 2006).

5.5 Lazy Methods

Lazy methods have come to form an important part of the multi-label literature.

One of the first lazy multi-label methods was the `MLkNN` method (Zhang and Zhou, 2007a) which adapts the k-Nearest Neighbor (kNN) algorithm for multi-label classification using Bayesian relevance. For each test instance $\boldsymbol{x}$, `MLkNN` takes the k nearest examples, $(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_k, \boldsymbol{y}_k)$, and counts the number of occurrences of each label in this neighborhood (i.e. the relevances across all $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k$) and combines this count with the prior probabilities of each label in the training data. The authors recommend setting $k = 10$, noting that the number of nearest neighbors has little effect on performance. `MLkNN` has become a very well-known method in the literature, and we compare with it in experimental evaluation.

I'd would say that there first stage is identical, but that they process the neighbourhood differently. And with regard to the neighbourhood both are clearly BR-like (unless I missed something about MLkNN), but one is more sophisticated than the other.

Spyromitros et al. (2008) review various lazy methods for multi-label learning, and present `BRkNN`, which is essentially identical to `MLkNN` in its first stage, but processes the label counts in the neighborhood without regard for prior probabilities. Additionally, they produce two extensions to eliminate the case where empty sets are predicted. This is relevant to `BR` because `BR` will produce an empty set $|\boldsymbol{y}| = 0$ for any test instance for which all its binary models predict 0 (here it is suggested that such a classification should be treated as erroneous). Upon encountering this condition, one extension (a) selects the label with the highest confidence and a second alternative extension (b) outputs the $[s]$ nearest labels where s is the average size of the label sets of the k nearest neighbors. Unsurprisingly, extension

(a) provides best results for *Scene*, whereas extension (b) provides best results for *Yeast*, due to the label cardinality composition of these respective datasets (LCard($Scene$) ≈ 1.0, LCard($Yeast$) > 4.0).

In the real world, forcing a classification so that $|\boldsymbol{y}| > 0$ is not always a good idea, in particular when active learning is used, or wherever empty predictions are preferred over poor predictions. Neither of these extensions would have any effect under AU($\overline{\text{PRC}}$) or our LOG-LOSS evaluation measure.

Relatively recently Cheng and Hüllermeier (2009) expanded on the work on lazy classification and introduced `IBLR`: a method which combines *instance-based learning* and *logistic regression*. `IBLR` takes label correlations into account by using the labels of neighboring examples as extra attributes in a meta logistic regression scheme. This secondary step is, in effect, a specialisation of the stacking procedure of the `MBR` paradigm.

`MLAC` (Veloso et al., 2007) is an *associative classification* approach that is also lazy, since it only completes the training process with respect to each test-instance as the test instance is given. It introduces multi-label *class association rules* as a way to model label correlations and dependencies among labels.

5.6 Probabilistic methods

A generative mixture model for multi-label text categorisation is proposed in (McCallum, 1999), which assumes that each document instance is generated by a mixture of (i.e. potentially multiple) single-label documents. This work employs naive Bayes with the expectation-maximisation (EM) algorithm to

estimate the model and mixture parameters, i.e. word distributions and the weights for each label.

Ueda and Saito (2002) present a similar work employing generative mixture models. Probabilistic methods like these explicitly model label correlations in a comparable fashion to LC. They therefore do not require a threshold or $|\hat{\boldsymbol{y}}|$-predicting function (like RT- and PW-based classification schemes—Section 3.2 discussed such functions). We note, however, that these methods are almost invariably applied to text-categorisation. In Section 2.3 we mentioned that an example labelled {A,B} is not necessarily a mixture of A-examples and B-examples, although this *is* often the case in text categorisation, and perhaps this is why the mixture model methods focus on this domain, and have not really been applied to other domains such as image classification.

Streich and Buhmann (2008) present a generative method where the contribution of each label to a given instance is estimated. Ghamrawi and McCallum (2005) present two models which are trained using data in which label co-occurrences have been captured by conditional probability models. Both models can capture arbitrary label-label relationships as additional attributes, and the second model also parameterises relationships between labels and attributes (i.e. words, in this case).

In a related approach, Sun et al. (2008) use a hypergraph method to model label correlations which, despite a proposed approximate formulation, induces high computational complexity.

Vilar and José (2004) work with multinomial naive Bayes on text data, using a thresholding strategy to bipartition a ranking so as to produce a label-

set prediction for each test instance. They also note that their thresholding approach can be applied to the BR method, although its application to a single model showed better results.

5.7 Neural Networks and Support Vector Machines

Section 5.2 reviewed the use of simple perceptrons as a base classifier for several PW approaches (for example CLR, MLPP, and DMLPP). There also exist several algorithm adaptations of neural models for multi-label classification.

The *multi-label multi-class perceptron* (MMP) (Crammer and Singer, 2003) is a BR-based ranking algorithm adapted from a perceptron model. Like most other multi-label neural-network approaches, the authors specifically study ranking quality on text collections. MMP is an online algorithm that works in rounds and can take a variety of ranking loss functions. MMP improves on BR, where BR is used for ranking with perceptrons as the base classifier, although this is not necessarily surprising because BR is not known for its ranking ability.

BP-MLL (Zhang and Zhou, 2006) is a neural network adaptation, which adapts the error function of a back-propagation algorithm so as to account for multiple labels in the learning process. Oliveira et al. (2008) adapt a probabilistic neural network algorithm for multi-label classification. Sapozhnikova (2009) modifies *adaptive resonance theory* neural networks for multi-label classification by encoding associations between input and target vectors.

SVMs are well known for their classification power, and as Section 4.4

showed, often perform very well in problem transformation contexts. There have also been specific adaptations.

`Rank-SVM` (Elisseeff and Weston, 2001) uses kernel SVM as a ranking function, and attempts to minimize ranking loss while maintaining a large margin. Ranking loss is the percentage of label pairs which are ordered incorrectly. A threshold function can be applied to the ranking to obtain a label-set classification and, more recently, Jiang et al. (2008) applied the CLR label-calibration scheme to this method, however the main focus of `Rank-SVM` is a ranking. Although showing superior performance to the aforementioned BoosTexter system in a ranking evaluation, the computational complexity of `Rank-SVM` ($O(N^2L)$ in terms of time) is too high for many datasets, as noticed by Petrovskiy (2006).

Tang et al. (2009) adapt a *one-vs-one* multi-class linear SVM implementation (as opposed to the *one-vs-rest* implementation of `SMO`) to produce a multi-label ranking as a vector of confidence scores, where one score pertains to each label. They review various threshold calibration methods which could be applied to the confidence outputs to produce a label-set prediction, but decide on a multi-class meta method to predict each $|\hat{\boldsymbol{y}}_i|$ i.e. the number of top labels from the ranking which form the predicted set for each instance. They experiment with training the meta method on both the scores and just the ranking.

In addition to the main contribution of their `MBR` method, Godbole and Sarawagi (2004) describe an SVM-specific modification, wherein training examples that are close to the hyperplane are removed.

5.8 Other Methods

`MMAC` (Thabtah et al., 2004) uses an associative classification method for multi-label classification. `MMAC` learns new rules iteratively until it has covered all training examples, then merges rules with similar preconditions into single rules. The method is related to the `LC` method where labels are treated in combinations, but deal with rankings for evaluation. `MuLAM` (Chan and Freitas, 2006) details another rule-based adaptation method, where the authors adapt an earlier single-label version of their ant-colony algorithm to work in a multi-label bioinformatics domain.

The `InsDiff` method (Zhang and Zhou, 2007b) computes a prototype vector for each label by averaging all the instances that are associated with that label. Each training example is then transformed into a bag of L examples (one for each label), where each instance in this bag is the difference between the original instance and the jth prototype vector. In its second stage, `InsDiff` learns from the transformed training set. This approach embraces the multi-label *multi-instance* paradigm.

5.9 Efficient Multi-label Classification

In this section, we summarise the methods in the literature with respect to their scalability, and tractability on large datasets.

First, let us quickly review the general method of artificial hierarchies, as presented in (Tsoumakas et al., 2008) and discussed in Section 2.5. This method decomposes a multi-label classification problem into a local-hierarchical arrangement, where each classifier is concerned with a subset of the label set,

as a way of reducing time complexity. Although such methods can reduce running times as compared with regular flat classification, they do so at a cost of predictive performance and often require significant memory-space overhead. On the other hand, since these methods are not specific to any particular classification scheme, our methods can employ them whenever desired, although this is not a focus of this thesis. Using *predefined* hierarchical structure relies on hierarchically-arranged datasets and success in this area has mainly been specific to algorithm adaptations in the bioinformatics domain (Vens et al., 2008; Kiritchenko, 2005; Barutcuoglu et al., 2006) and are thus not generally applicable.

Many approaches reviewed in this chapter scale to large data sources by using very efficient classifiers, either in an algorithm adaptation or as base classifiers in problem transformation. Naive Bayes is a common choice for this context, used both as an algorithm adaptation, for example (McCallum, 1999; Ueda and Saito, 2002), and problem transformation, for example (Tsoumakas et al., 2008). Simple perceptrons have also been popular (Fürnkranz et al., 2008; Mencía and Fürnkranz, 2008; Loza Mencía and Fürnkranz, 2008), especially in `PW` approaches. Most probabilistic and lightweight neural approaches tend to be focussed on document ranking for text categorisation and as such, are more specialist methods which have so far not yet been shown to be generally applicable. In most cases, lightweight classifiers are used because the learning task would not otherwise have been tractable; these approaches are certainly efficient but—we argue—not general, as they cannot adapt easily to different data. This thesis focusses on methods of high predictive performance across a range of data, as well as efficiency.

Decision tree adaptations are generally very efficient because a single decision tree model can carry out multi-label predictions. However, such models are usually specific to biological domains, as in (Clare and King, 2001), and designed to perform on a hierarchical structure, as in (Vens et al., 2008). In the general multi-label context, other methods have been approached instead. We have previously discussed the relative inflexibility of algorithm adaptation methods when it comes to finding scalable and generally applicable methods.

More intensive classification schemes like `Rank-SVM` and in BoosTexter have so far proved less scalable. Tang et al. (2009) provide a relatively efficient SVM-based scheme which scales to training sets of $N = 12,300$ and $L = 23$, although these statistics refer to two different training sets, both of which are relatively small compared with many of the datasets in our collection (listed in Table 2.1) upon which we later test our problem transformation methods (in Chapters 6 and 7). We also note that this method requires training a multi-class meta method to give the size of each label-set prediction.

Most methods which scale to large datasets are based upon `BR`. For text data where L is large, Ráez et al. (2004) found that they could ignore rare labels altogether (i.e. removed binary models) to achieve increased performance in imbalanced text-categorisation problems. (Tsoumakas et al., 2009a) and (Yan et al., 2007) share binary models between labels and the latter also takes subsets of the feature set. These strategies are all generally applicable to `BR` and we will discuss them again in later chapters.

The following two chapters introduce our novel methods. In the experi-

mental evaluations of these chapters we will often refer to methods that have been described here.

Chapter 6
Pruned Sets

In (Read, 2008) we introduced the *pruned sets method*. This method improves on the well known label-combination method (LC), achieving a much reduced running time, and scalability to larger datasets, and better predictive performance. In (Read et al., 2008) we improved this method and introduced an *ensemble of pruned sets* which proved highly competitive against other methods in the literature. This chapter introduces further improvements to these methods and greatly expands the analysis and evaluation of their performance.

6.1 Issues with LC

By treating every label combination in the training data as a unique class label in a single-label problem, LC directly takes into account label correlations and, as a result, often performs much better than BR. However, a major disadvantage of LC is the number of class labels it must represent in its problem transformation: one for each distinct label set which exists in the training data (a worst case of $\text{MIN}(N, 2^L - 1)$ class labels). This has a

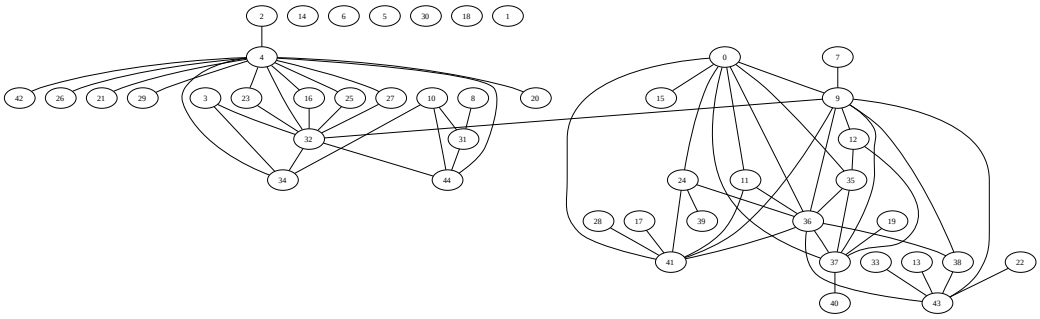
direct implication on complexity and limits of scalability, and additionally leads to over-fitting the training data, and a sparsity of training examples to class labels in the single-label transformation, since many label combinations are likely to occur very infrequently in the data (often pertaining to a single example).

We now review our *pruned sets* method, which tackles the disadvantages of `LC` by concerning itself with a simpler and more balanced representation of the data.

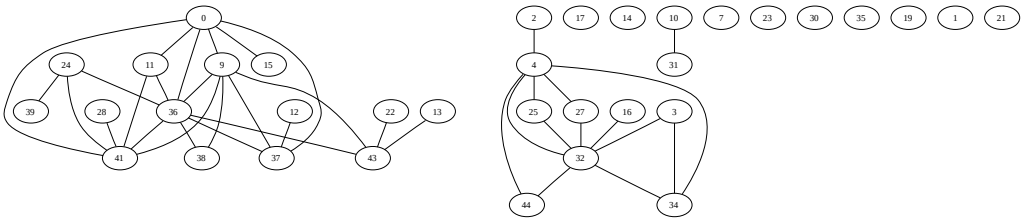
6.2 The Pruned Sets Method (`PS`)

Consider the graphs in Figure 6.1, which are derived from the *Medical* dataset. The nodes represent the labels of this dataset, and each edge represents a co-occurrence between two labels. In the initial graph, each edge represents at least one co-occurrence. In the second graph, each edge represents at least two co-occurrences, i.e. the examples where the label set only occurs once in the training data have been *pruned* from the training set. When all edges represent at least three co-occurrences, this leaves a relatively simple graph which still represents 92% of all examples. Such simplified representations are the basis for the *pruned sets* method (`PS`).

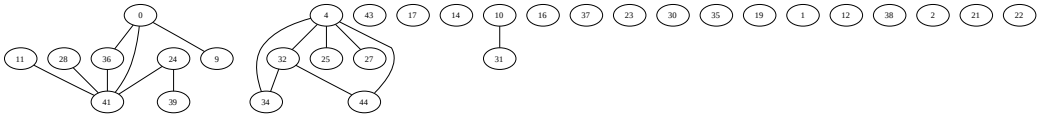
The *Medical* dataset is not a special case. Section 2.1 already discussed how multi-label datasets typically display *label skew* where a significant proportion of label sets occur relatively infrequently, while a small proportion of the label combinations occur very frequently. Figure 6.2 illustrates this across datasets by plotting the 'long tail' effect where label combinations are



(a) Each edge represents ≥ 1 co-occurrences (100% of examples)



(b) Each edge represents ≥ 2 co-occurrences (97% of examples)



(c) Each edge represents ≥ 3 co-occurrences (92% of examples)

Figure 6.1: Label co-occurrences in the *Medical* dataset.

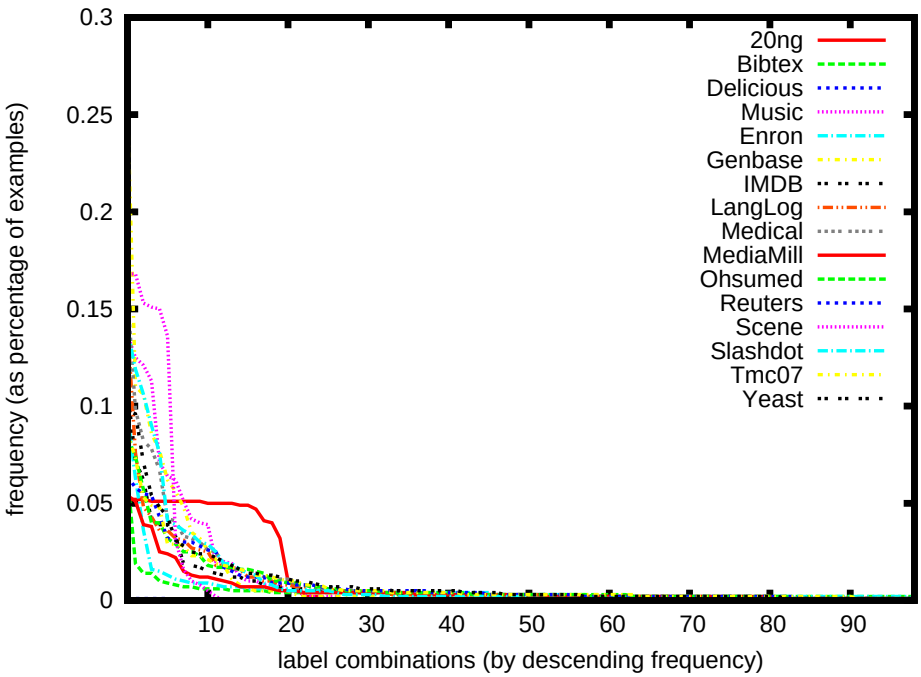


Figure 6.2: Label combinations are plotted by their frequency for datasets across the collection.

plotted by frequency in descending order. *20ng* is the exception to this rule because it was artificially compiled with 1000 examples associated with each label.

The primary motivation behind `PS` is to use the power of `LC`'s label-set-based paradigm to capitalise on the most important label relationships found within a multi-label dataset, but without succumbing to the same disadvantages of over-fitting and high running time suffered by `LC`.

The `PS` algorithm contains two important steps: a *pruning* step and a label-set *subsampling* step. The pruning step removes infrequently occurring label sets from the training data. This removes unnecessary and detrimental complexity from the `LC`-transformed data by reducing the number of class la-

bels. However, PS does not simply discard pruned examples, on the grounds that considerable information can be lost. Rather, PS subsamples the label sets of these examples for label *sub*sets[1] which occur more frequently in the training data. It then attaches these label sets to the instance of the example, thus creating new examples, and reintroduces these examples into the training. In this way, information from the pruned examples is conserved without the time complexity, since infrequent label sets are much more influential on an LC-classifier's running time than the number of training examples. Since PS focusses only on the core label sets, the tendency to overfit the training data is also greatly reduced.

The pruning and subsampling steps are each influenced by a parameter. PS takes these parameters and a set of training examples, and returns a modified training set upon which an LC-classifier is trained (actually, any method can be trained on the transformed data but only the LC paradigm benefits greatly from the reduced number of label sets). The PS algorithm is outlined in Algorithm 6.1. Figure 6.3 shows the flow of training examples. Note that $\textsc{count}(\boldsymbol{y}, \mathcal{D})$ simply returns the number of times the label set $\boldsymbol{y}$ occurs in the training data $\mathcal{D}$. The SUBSAMPLE function will be discussed in Section 6.2.2.

6.2.1 The pruning parameter (p)

The pruning parameter (p) determines how much pruning to do, where $p = 0$ defaults to the functionality of LC, $p = 1$ prunes all examples where the label

[1] As forewarned in Section 1.4, we will use set notation in this chapter: $(\boldsymbol{x}, \boldsymbol{y})$ is a training example; $\boldsymbol{y}' \subset \boldsymbol{y}$ is a label *sub*set, and also a label set if $(\boldsymbol{x}, \boldsymbol{y}')$ for some $\boldsymbol{x}$.

Algorithm 6.1 The PS algorithm; given a training set $\mathcal{D}$, pruning parameter p, and label-set subsampling parameter n.

PS$(\mathcal{D}, p, n)$
1 $\mathcal{D}' \leftarrow \mathcal{D}$
2 $\mathcal{D}_P \leftarrow \{\}$
3 $\triangleright$ Pruning Step
4 **for** $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$:
5 **do if** COUNT$(\boldsymbol{y}, \mathcal{D}) \leq p$:
6 **then** $\mathcal{D}_P \leftarrow \mathcal{D}_P \cup (\boldsymbol{x}, \boldsymbol{y})$
7 $\mathcal{D}' \leftarrow \mathcal{D}' \setminus (\boldsymbol{x}, \boldsymbol{y})$
8 $\triangleright$ Subsampling Step
9 **for** $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_P$:
10 **do for** $\boldsymbol{y}' \in$ SUBSAMPLE$(\boldsymbol{y}, \mathcal{D}', n)$:
11 **do** $\mathcal{D}' \leftarrow \mathcal{D}' \cup (\boldsymbol{x}, \boldsymbol{y}')$
12 **return** $\mathcal{D}'$

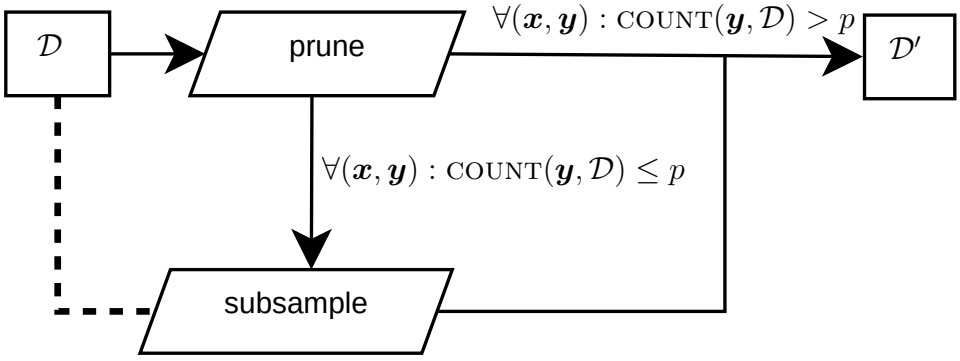


Figure 6.3: The flow of training examples in the PS algorithm.

set is unique, $p = 2$ prunes all examples which occur at most twice, and so on. $p \geq N$ would result in the entire training set being pruned.

The parameter p is closely comparable to the *minimum support count* known to *itemset mining*, where every label set could be considered an itemset. In fact, we will often refer to p by this term, although note that the minimum support count is actually equivalent to $p + 1$ as opposed to p in our notation.

6.2.2 The label-set subsampling parameter (n)

The label set of each pruned example becomes a candidate for label-set subsampling. The subsampling parameter n influences which, and how many subsets, are drawn from these label sets.

PS subsamples the label sets of pruned examples to create examples which *do* meet the pruning criterion. For example, suppose $(\boldsymbol{x}, \boldsymbol{y})$ where an image $\boldsymbol{x}$ is associated with a rare label combination $\boldsymbol{y} = \{\texttt{sunset}, \texttt{mountain}, \texttt{field}\}$. We can subsample this label set for more frequently occurring combinations, like $\boldsymbol{y}'_1 = \{\texttt{sunset}, \texttt{mountain}\}$ and $\boldsymbol{y}'_2 = \{\texttt{sunset}, \texttt{field}\}$, and associate these sets with $\boldsymbol{x}$ instead, i.e. to create $(\boldsymbol{x}, \boldsymbol{y}'_1)$ and $(\boldsymbol{x}, \boldsymbol{y}'_2)$ to replace $(\boldsymbol{x}, \boldsymbol{y})$. In this way, PS preserves information from the example without the associated complexity of an extra class label for them. However, $\boldsymbol{y}'_3 = \{\texttt{mountain}\}$ and $\boldsymbol{y}'_4 = \{\texttt{field}\}$ are also possible subsets. PS has to decide which are the best subsets to subsample.

PS only considers subsets which meet the original pruning criterion (any $\boldsymbol{y}' \subset \boldsymbol{y}$ where $\text{COUNT}(\boldsymbol{y}', \mathcal{D}) > p$). However, using all such subsets from each label set will introduce unnecessary complexity in the form of new exam-

ples and potentially hamper predictive performance. A heuristic is needed. In (Read et al., 2008) we used different configurable *subsampling strategies*. We have since discovered that by initially ranking all allowable subsets, the subsampling process can be simplified: the top n possible label subsets are used to create new examples for the training set, where n is the configurable parameter for this process.

Since we wish to preserve as much information as possible about the relationships within each label set, larger subsets should be given a higher priority than smaller subsets, since larger sets retain more label-correlation information. Also, in the interest of maintaining the focus on the core combinations, label sets with a higher frequency in the training data should have a higher priority than less frequently occurring sets. We can embody these two rules in a *compare* function for any off-the-shelf sorting algorithm; see the COMPARE$(\cdot,\cdot)$ function in Algorithm 6.3. Therefore, given a label set $\boldsymbol{y}$, all the subsampling routine has to do is generate all label subsets which meet the original pruning criterion, sort them in order of priority, and return the top n sets: see the SUBSAMPLE process in Algorithm 6.2. Algorithm 6.1 showed how these n label sets are introduced into the training set with a copy of the original instance.

Algorithm 6.2 PS's subsampling function.

SUBSAMPLE$(\boldsymbol{y}, \mathcal{D}', n)$
1 $\triangleright$ an ordered set of all d subsets of $\boldsymbol{y}$ which occur in $\mathcal{D}'$
2 $\boldsymbol{q} = (q_1, \ldots, q_d) \leftarrow \{\boldsymbol{y}' | \boldsymbol{y}' \subset \boldsymbol{y}, \exists \boldsymbol{x} : (\boldsymbol{x}, \boldsymbol{y}') \in \mathcal{D}'\}$
3 $\boldsymbol{q} \leftarrow$ SORT$(\boldsymbol{q},$ COMPARE$(\cdot,\cdot))$
4 **return** $\{q_1, q_2, \ldots, q_{\min(n,d)}\}$

Algorithm 6.3 The compare function used for the SORT routine in the SUBSAMPLING function.

```
COMPARE(y_A, y_B)
1  if |y_A| > |y_B|
2      then return 1
3  if |y_B| > |y_A|
4      then return −1
5  if COUNT(y_A, D) > COUNT(y_B, D)
6      then return 1
7  if COUNT(y_B, D) > COUNT(y_A, D)
8      then return −1
9  return 0
```

6.3 Parameter Configuration

The configuration of the p and n parameters clearly affects the performance of `PS`. The aim of `PS` is to be tractable for larger problems (where `LC` is not), but also compete with the predictive power of `LC`. We thus configure the parameters in a manner most likely to result in the highest possible predictive performance, while at the same time expecting a reduced running time over `LC`.

Clearly, the pruning parameter p offers a trade-off between predictive performance and training time. Intuitively, more pruning (higher p) reduces complexity, but too much pruning will have a detrimental effect on predictive performance. This is best viewed under empirical analysis: Figure 6.4 shows the relationship between p and ACCURACY, and Figure 6.5 shows the relationship between p and training time. At $p = 0$, `PS` is equivalent to

standard LC. ACCURACY is relatively stable for low values of p and begins to decline under higher values as more pruning is carried out; although this degradation is often only gradual or even negligible. Training time is invariably reduced dramatically; particularly observable on *TMC2007*, *Ohsumed* and *Enron*, which are very complex for low values of p, and did not finish at $p = 0$ (where LC $\equiv$ PS). They become quickly tractable under even smaller values of p. Values $1 \leq p \leq 5$ generally achieve the highest ACCURACY.

We have so far considered pruning in isolation (without the label-set subsampling process; i.e. $n = 0$) and the plots should not be used to gauge the absolute predictive performance of PS nor relative performance versus LC. Let us now look at the performance of PS when the subsampling process is turned on (i.e. $n > 0$). Table 6.1 shows the effect on ACCURACY under $p = 1$ and $p = 3$. Figure 6.6 plots the effect of a range of n on training times for $p = 3$.

The subsampling function is indeed able to improve the predictive performance of PS, with the exceptions of *Yeast*, *TMC2007*, and *Enron* (we note that in these cases $\text{LC}\textsc{ard}(\mathcal{D})$ is high). Label-set subsampling made no difference to ACCURACY on *Genbase*: analysis revealed that although subsampling occurred, it was too infrequent to affect predictions. Training time increases at most linearly and generally insignificantly. The maximum effective range of n is closely bound to $\text{LC}\textsc{ard}(\mathcal{D})$, which is itself invariably limited. From this analysis we see that $n = 1$ to 5 is a good range for most datasets and $n = 0$ where $\text{LC}\textsc{ard}(\mathcal{D})$ is high, say > 2.0.

In very rare cases, where the majority of label sets are unique and the label scheme is irregular, the subsampling process will begin to break down,

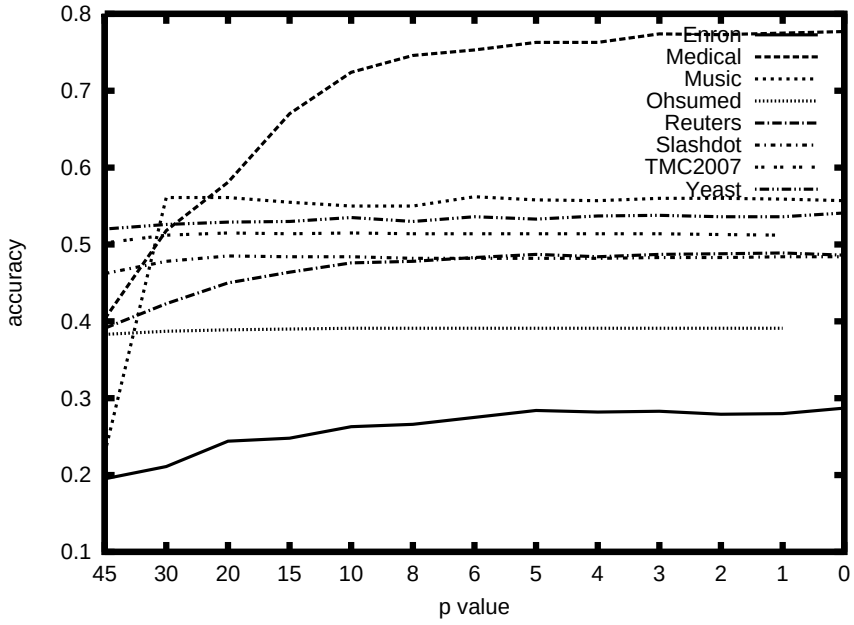


Figure 6.4: The effect of the p parameter on ACCURACY; $n = 0$; `SMO` is the single-label base classifier.

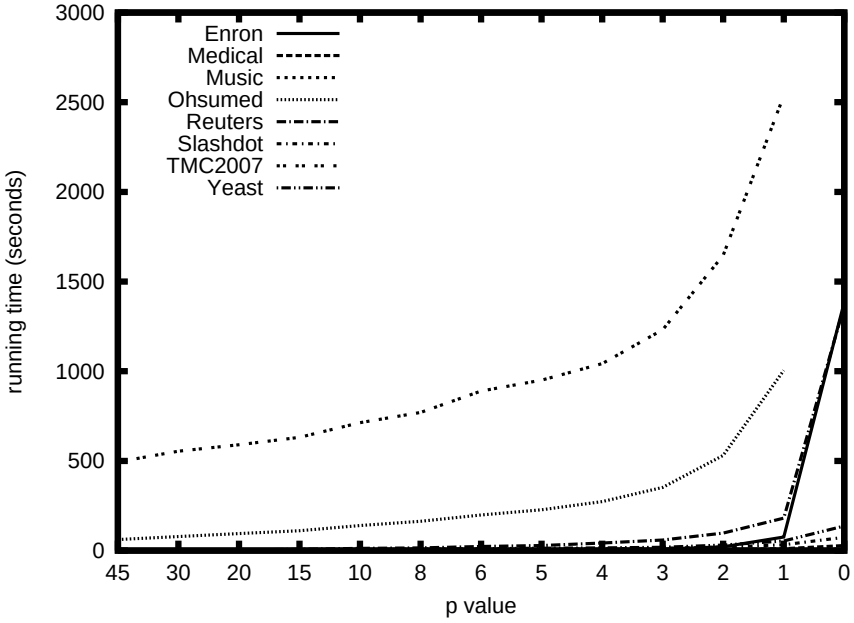


Figure 6.5: The effect of the p parameter on training time; $n = 0$; `SMO` is the single-label base classifier.

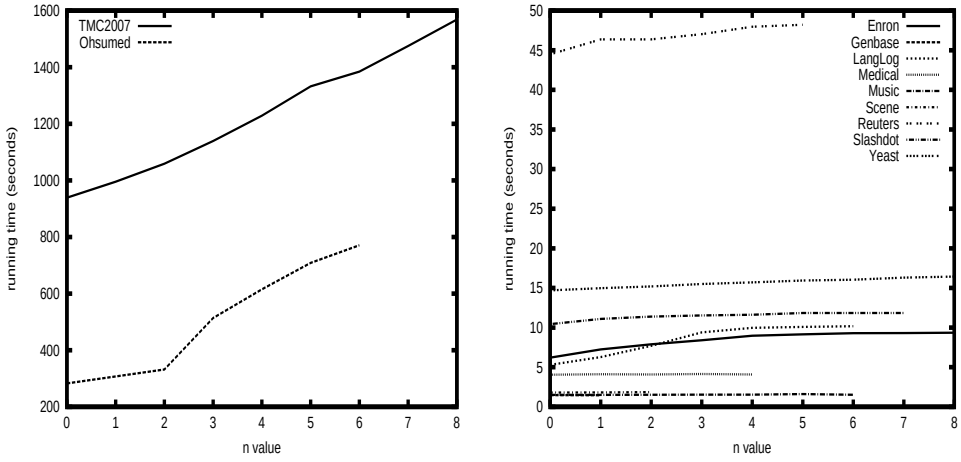


Figure 6.6: The effect of the n parameter on training time; $p = 3$; `SMO` is used as the single-label classifier.

Table 6.1: The effect of the n parameter on ACCURACY, where $p = 1$ (top) and $p = 3$ (bottom).

$p = 1$

n	20ng	Enron	Genb.	L.Log	Med.	Music	Ohsu.	Reut.	Slash.	TMC.	Yeast
0	0.587	0.280	0.989	0.131	0.775	0.559	0.391	0.066	0.484	**0.512**	**0.536**
1	0.588	**0.309**		**0.137**	0.770	0.564	0.393	0.490	0.480	0.508	0.532
2	**0.589**	0.292		0.126	**0.786**	**0.565**	0.392	**0.493**	0.478	0.510	0.526
3		0.287		0.113		0.564	**0.394**		0.477	0.507	0.533
4		0.284		0.110		0.562	0.390		0.483	0.509	0.526
5		0.263		0.111			0.389		0.484	0.510	0.529
6		0.272		0.110			0.391		**0.486**	0.509	0.531
7		0.259									0.527
8		0.254									0.530

$p = 3$

n	20ng	Enron	Genb.	L.Log	Med.	Music	Ohsu.	Reut.	Slash.	TMC.	Yeast
0	0.588	**0.283**	0.980	0.130	0.774	0.360	0.391	0.487	0.483	**0.514**	**0.538**
1	0.589	0.282		**0.137**	0.772	0.352	**0.397**	0.491	0.477	0.508	0.533
2	**0.591**	0.266		0.123	**0.784**	0.353	0.394	0.494	0.487	0.510	0.525
3	0.589	0.270		0.106	0.783	0.360	**0.397**	**0.495**	0.488	0.508	0.524
4	0.590	0.258		0.096		0.371	0.394	0.489	0.487	0.510	0.521
5	**0.591**	0.253		0.098		**0.373**	0.392		**0.489**	0.508	0.520
6		0.247							0.488	0.510	0.514
7		0.244								0.506	0.509
8		0.241									0.505

since too many label sets are pruned away and subsampling may not be possible. Such cases usually indicate an extraordinary lack of regularity in the data, and the dataset in question may not be interesting as a multi-label problem (or `BR` could be safely used instead). For this reason, we believe that such worst cases generally need not be seriously considered, although in Section 6.5 we describe a modification to the SUBSAMPLE method to deal with this scenario.

Both the p and n parameters can be tuned using internal cross validation on the training set or a hold out set, as we reported in (Read et al., 2008). However, as Section 6.9 shows, with the simplified parameterisation of the subsampling mechanism presented in this chapter, ad hoc values can suffice

and, in an ensemble scheme (presented in Section 6.7), good results can be obtained by selecting from a range of values for each ensemble member.

6.4 Time Complexity

It is difficult to use the theoretical time complexity of PS as a guide to actual performance, since PS's worst-case scenarios (such as where every label set is unique) are unlikely to occur in practice. Generally, we can say that although the theoretical worst-case time complexity is equivalent to LC, the real-world average time complexity is expected to be much better.

The previous sections considered PS's parameter values on time complexity individually. Let us now consider the relationship they have on each other and the effect they have together on overall complexity.

The most complex configuration for PS involves low values of p (less pruning) and high values of n (more subsampling). However, in practice these worst cases are inversely dependent on each other. Less pruning implies fewer examples for subsampling, and more subsampling is only possible when more pruning is carried out. The value of p has the strongest effect on time complexity. We can say that when $p = 1$, a PS transformation will have $\text{PUNIQ}(\mathcal{D})$ fewer class labels than an LC transformation and there will be at most $n \times N_P$ more examples where N_P is the number of pruned examples.

6.5 PS as an Itemset Problem

It is easy to notice a connection between PS and *itemset mining*, when label sets are considered as itemsets. We have already pointed out how the p parameter resembles the support count of itemset problems.

It is possible to modify PS's subsampling process so that it generates and considers all *frequent itemsets* rather than just frequently occurring label sets. Let us quickly clarify some terminology:

- $\boldsymbol{y}$ is a *frequently occurring label set* if it meets the minimum support count; i.e. $\textsc{count}(\boldsymbol{y}, \mathcal{D}) > p$.
- $\boldsymbol{y}'$ is a *frequent itemset* if it is a frequently occurring label set *or* is a subset of such a set; i.e. $\boldsymbol{y}' \subseteq \boldsymbol{y}$.

For example, suppose: $\{a, b, c\}$ and $\{b, c, d\}$ have a count of 1, thus not meeting a minimum support of 2 ($p = 1$). These sets are therefore *not* frequent. However, $\{b\}, \{c\}$ and $\{b, c\}$ *are* frequent.

At first glance this seems an appealing extension to PS, since PS would then not be restricted to only the label combinations in the training data, and may therefore reduce the effects of overfitting. However, it turns out that such a process detracts from the advantages of the PS paradigm in two crucial ways: by not modelling label combinations exactly as they are found in the data, and by increasing the complexity of the training process.

We confirmed that in practice, for almost all datasets, applying itemset mining techniques to PS increases computational complexity without benefiting predictive performance. However, there are exceptions: where the percentage of label sets that are unique ($\textsc{PUniq}(\mathcal{D})$) *and* label cardinality

(LCard($\mathcal{D}$)) are both high, PS's subsampling process may begin to break down. This is the case for *Enron*. An itemset-inspired modification replaces line 2 of the SUBSAMPLE function (Algorithm 6.2) with:
$\boldsymbol{q} \leftarrow (q_1, \ldots, q_d) = S$, where S is the set of all frequent itemsets generated from $\boldsymbol{y}$ with respect to $\mathcal{D}'$.

Given this modification, the worst case only holds where every label set in the training data is unique *and* no frequent itemsets exist. We are not aware of any such dataset in the real world.

Although PS is generally better without itemset mining techniques applied to it, it must be nevertheless acknowledged that, on account of being restricted to only the label combinations in the training set, PS may encounter problems with over-fitting when dataset labelling is irregular. Alternatives to itemsets are presented in the following sections.

6.6 PS with a Threshold Function

PS can improve the predictive performance of LC and run much faster. However, it still retains one of LC's cited flaws: it is only able to classify new examples with label-subsets observed previously in the training data. This makes it vulnerable to datasets like *Enron* with very irregular labelling. To overcome this problem, PS needs to be able to form new label sets at classification time.

In (Read, 2008) we presented a modification to PS which can form new label sets at classification time by using a *threshold function*: PS^t (threshold functions were reviewed in Section 3.2). Unlike itemset mining strategies,

this adaptation does not affect the time complexity at training time. The classification process of $\texttt{PS}^t$ is outlined in Figure 6.4 and an example is illustrated in Figure 6.7 which utilises the threshold function f_t of Section 3.2.

$\texttt{PS}^t$ requires that the base classifier, which predicts label combinations (as single labels), can provide confidence outputs for each of these combinations. Let these confidence outputs be a vector $\hat{\boldsymbol{w}}' \in \mathbb{R}^{L'}$ where $\hat{w}'_k$ is the confidence associated with the kth label in $\mathcal{Y}'$: $\mathcal{Y}' = \{\boldsymbol{y}'_k | k = 1, \ldots, L'\}$ is the set of class labels in the transformed single-label problem, each of which is a $\boldsymbol{y}$-type labelset (vector), i.e. a label combination. Note that we are using prime ($'$) to denote variables relating to the single-label space.

Algorithm 6.4 The prediction algorithm of $\texttt{PS}^t$.

```
CLASSIFY(x, h, f_t)
1  global Y' ▷ the transformed single-label space
2  w' = h(x) ▷ confidence outputs R^{L'} with respect to Y'
3  ŵ ← R^L ▷ confidence outputs for the multi-label space
4  for k ← 1, ..., L'
5        do
6           y' ← Y'_k
7           ŵ_j ← Σ_{j=0}^{L} y'_j w'_k
8  return f_t(ŵ)
```

The advantage of $\texttt{PS}^t$ is that it can form new label sets at classification time, and in this way handles irregular labelling, where many test instances are associated with label combinations which have not been observed in the training set. A disadvantage, however, is the reliance on prediction confidence distributions of the base classifier. Different base classifiers give different

$\mathcal{Y}' \downarrow / \mathcal{Y} \rightarrow$	$\{$	1	2	3	4	5	6	$\}$	$\downarrow \hat{\boldsymbol{w}}' = h(\boldsymbol{x})'$
$\boldsymbol{y}_1 =$	$[$	0	0	1	1	0	0	$]$	$\hat{w}'_1 = 0.8$
$\boldsymbol{y}_2 =$	$[$	1	0	0	1	0	1	$]$	$\hat{w}'_2 = 0.4$
$\boldsymbol{y}_3 =$	$[$	0	1	0	0	0	0	$]$	$\hat{w}'_3 = 0.0$
$\boldsymbol{y}_4 =$	$[$	0	0	1	0	0	1	$]$	$\hat{w}'_4 = 0.6$
$\boldsymbol{y}_5 =$	$[$	1	1	0	0	0	0	$]$	$\hat{w}'_5 = 0.0$
$\hat{\boldsymbol{w}} =$	$[$	0.40	0.00	1.40	1.20	0.00	1.00	$]$	
$\overline{\boldsymbol{w}} =$	$[$	0.10	0.00	0.35	0.30	0.00	0.25	$]$	(normalised)
$\hat{\boldsymbol{y}} = f_t(\overline{\boldsymbol{w}}) =$	$[$	0	0	1	1	0	1	$]$	$t = 0.2$

Figure 6.7: An example of $\texttt{PS}^t$'s prediction for given values of $\mathcal{Y}$,$\mathcal{Y}'$,t, and $\hat{\boldsymbol{w}}'$ for a test instance $\boldsymbol{x}$.

confidence distributions, and some may not provide a quality distribution, or even a distribution at all. In the next section we discuss an ensemble method which can also form new combinations at classification time, but does not rely on base classifier prediction confidence distributions to do so.

6.7 Ensembles of Pruned Sets (EPS)

In this section we present *ensembles of pruned sets* (EPS), a method which employs PS in an ensemble framework, and uses a voting scheme to produce the prediction confidences instead of the prediction confidences of the single-label classifier like $\texttt{PS}^t$. Whereas $\texttt{PS}^t$ can perform well in particular domains with base classifiers like naive Bayes (which provide smooth confidence outputs), EPS provides a powerful and general framework. This comes at a cost in performance but is justified by the improvement in predictive performance.

PS is particularly suited to an ensemble due to its fast build times. And,

if each PS classifier models different core label sets of the training set, then an ensemble can form new label combinations at prediction time by compiling label-set predictions via a simple voting procedure and a threshold function. The ensemble thus overcomes the disadvantage of standalone PS in a more general and powerful way than $\texttt{PS}^t$, and counters any over-fitting effects of the pruning process. Ensembles are well known for their effect of increasing overall predictive performance, and allowing parallelism. Moreover, ensembles are inherently scalable: a larger ensemble (more models/iterations) may achieve better predictions and a smaller ensemble results in lower running-times.

EPS's training algorithm is outlined in Algorithm 6.5. This algorithm can be used with any multi-label-capable classifier, but works particularly well with PS. Many methods work well under a *bagging* ensemble (Breiman, 1996), where each model is trained on a bag of N examples that are selected with replacement from the training set $\mathcal{D}$. However, PS's label-set subsampling function already duplicates instances and therefore further duplications of examples in the ensemble scheme are not necessary. Thus, our ensemble scheme takes simple subsets of the training set *without replacement* (we found subsets of around 62% work best). In the context of using PS models, our ensemble scheme is both faster and provides superior predictive performance than bagging. Experimental results in Tables 6.5, 6.6 and 6.7 in the following section show this.

The voting scheme used for classification is detailed in Algorithm 6.6. It is also generic. It uses a threshold function to draw a label set from the confidence outputs (which are votes).

Algorithm 6.5 A generic ensemble training algorithm for multi-label classifiers.

TRAIN($\mathcal{D}, h, m, c$)

```
1  ▷ D training set
2  ▷ h a multi-label classifier
3  ▷ m number of models
4  ▷ c percentage of D for each model
5  for k ← 1 to m
6       do h_k ← h.COPY()
7          ▷ k is used as random seed
8          D ← RANDOMIZE(D, k)
9          D' ← c% of D
10         h_k.TRAIN(D')
```

Algorithm 6.6 A generic ensemble classification algorithm for multi-label classifiers.

CLASSIFY($\boldsymbol{x}, h_{1,\ldots,m}, f_t$)

```
1  ▷ x test instance
2  ▷ h_{1,...,m} multi-label classifiers
3  ▷ f_t threshold function
4  ŵ ← R^L
5  for k ← 1 to m
6       do ŷ = h_k(x)
7          for j = 1 to L
8               do ŵ_j ← ŵ_j + ŷ_j
9  ▷ multi-label prediction via threshold function
10 return f_t(ŵ)
```

6.8 Related Work

LC has not frequently been employed directly in the literature; likely due to its computational complexity, which makes it intractable on most datasets. Boutell et al. (2004) use an LC method on *Scene*, although this dataset is quite small.

Algorithm adaptation methods, such as the mixture model by McCallum (1999) and the association rule learner by Thabtah et al. (2004), have embraced the LC paradigm by treating combinations of labels as single labels. RAkEL (Tsoumakas and Vlahavas, 2007) is certainly the most well known LC-based problem transformation approach. RAkEL trains LC methods on the subsets of the label set of size k, thus incurring complexity quadratic with k rather than L. These LC methods are run in an ensemble. EPS is compared with RAkEL in the experiments that follow.

6.9 Experiments

In this section we compare PS methods to LC, and the ensemble version EPS to a variety of state-of-the-art methods. Section 3.4 detailed our typical experimental setup.

6.9.1 PS against LC

Initially we compare PS with the standard LC method directly. We consider two parameter configurations for PS (the reason behind the choice of 2.0 was explained in Section 6.3):

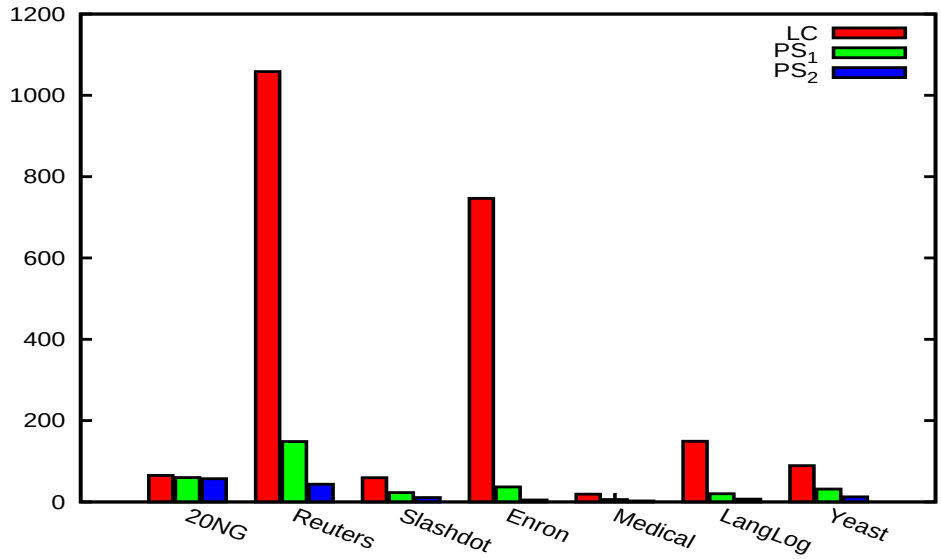
Model	Parameters
$\texttt{PS}_1$	$p = 1$, $n = \begin{cases} \text{if } (\text{LC}\textsc{Card}(\mathcal{D}) > 2.0) & 0 \\ \text{otherwise} & 3 \end{cases}$
$\texttt{PS}_2$	$p = 3$, $n =$ (same as above)

We also compare these versions of PS with the threshold extension ($\texttt{PS}^t$).

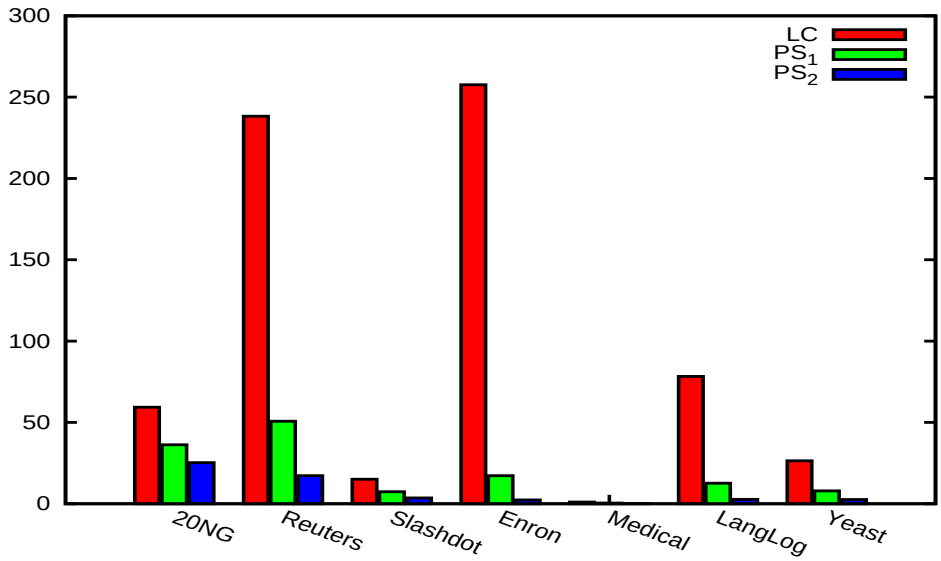
Table 6.2 displays the results of 5×2 cross validation for two predictive-performance measures with statistical significance indicated according to a corrected paired t-test. Table 6.3 displays running times, and Figure 6.8 plots a selection of these. SMO is the base classifier here but, since thresholding functions are dependent on the confidence output distributions of the base classifier, we also carry out this experiment using J48 and provide the results of predictive performance in Table 6.4 for comparison.

$\texttt{PS}_1$ and $\texttt{PS}_2$ show statistically significant gain over LC on three datasets under ACCURACY and on five datasets under HAMMING-LOSS. In the latter case, all statistically significant differences are in favour of PS except *Genbase* for $\texttt{PS}_2$. Comparatively, the $\texttt{PS}^t$ methods struggle where SMO is used, due to the relatively poor confidence predictions which this classifier is able to provide (since by default, this classifier does not fit logistic models to SVM outputs).

The experiment with J48 as the base classifier illustrates two points. Firstly, that the gains of PS over LC can be reproduced with base classifiers other than SMO—in fact the relative improvement in performance is even greater in this case. Secondly, it shows how the performance of $\texttt{PS}^t$ is relatively higher when smoother prediction confidences are provided. This



(a) Running times.



(b) Testing times.

Figure 6.8: Time measurements.

performance is particularly significant for the *Enron* dataset—as expected—, and *LangLog*. On both these datasets the $\texttt{PS}^t$ schemes are the best overall method in terms of ACCURACY, and also do well under HAMMING-LOSS.

Although the gains made by PS over LC could be viewed as modest in terms of predictive performance, the reduction in running time is very significant, especially on datasets like *Reuters*, *Enron*, and *LangLog*. These datasets all have relatively high $\text{LCARD}(\mathcal{D})$ and $\text{PUNIQ}(\mathcal{D})$ values, which indicate that many class labels are generated by an LC transformation. PS prunes infrequent sets (which would otherwise be class labels) out of the training data and is up to orders of magnitude faster.

We would expect PS to achieve relatively higher predictive performance for methods if the p and n parameters were tuned individually for each dataset, as we did in (Read et al., 2008) with internal cross validation. However, our main focus for predictive performance is EPS, where we invest computational complexity in ensemble iterations rather than fine-tuning parameters, so as to compete with state-of-the-art methods in the literature.

6.9.2 Comparing EPS with a bagging and a simple ensemble

To show the advantage of the simple subset ensemble as opposed to bagging in the context of PS, we compare the performance of $\texttt{PS}_2$ in 10 iterations in both schemes. Results are provided in Tables 6.5 (ACCURACY), 6.6 (EXACT-MATCH), and 6.7 (running time). Our ensemble scheme often achieves statistically significant predictive performance and, as expected, is generally around twice as fast: EPS with simple subsets of 62% of the training set sees

38% fewer training examples than EPS under bagging.

6.9.3 EPS against state-of-the-art methods

We compared EPS to a range of state-of-the-art methods: the competitive and well-known LC-based RAkEL, the PW-based CLR, and the kNN-adapted MLkNN, all of which were reviewed in Chapter 5—thus providing a variety of competing methods. We also included standard BR, a common base-line method in much of the multi-label literature.

Both EPS and RAkEL rely on parameter configurations. We created two configurations of each as follows:

Method Name	#Models	Model Parameters
$\texttt{EPS}_1$	$m = 10$	$p = 1$, $n = \{1—3\}$
$\texttt{EPS}_2$	$m = 50$	$p = \{1—5\}$, $n = \{1—3\}$
$\texttt{RAkEL}_1$	$m = 10$	$k = L/2$
$\texttt{RAkEL}_2$	$m = L \times 2$	$k = 3$

$\{\cdot—\cdot\}$ indicates that a number was chosen randomly from a range (inclusive)

We found these configurations to work well in terms of predictive performance, as did the authors of RAkEL for their method. Configurations '1' allow for more complexity to take into account label correlations, and configurations '2' allow for more iterations and a slightly more efficiently configured base method. In configuration '2', where EPS has an ensemble size of 50, selecting from a range of values for each PS model becomes appropriate: all models should capture the core label combinations, while some models will

provide relief in terms of computational complexity. The discussion and illustrations in Section 6.3 provide the intuition for these configurations of p and n. We used `SMO` for all problem transformation methods (which excludes kNN-adapted `MLkNN`).

Predictive performance is examined for four evaluation measures: ACCURACY (Table 6.8); EXACT-MATCH (Table 6.9); AU($\overline{\text{PRC}}$) (Table 6.10); F1-MACRO$^{\times L}$ (Table 6.11); and LOG-LOSS (Table 6.12) – all under 5×2 cross validation with significance under a corrected paired t-test indicated against $\texttt{EPS}_2$ (note that numbers are directly comparable with Table 6.2 involving standalone `PS`). Additionally, running times are displayed in Table 6.13. A missing result indicates *did not finish* within time and memory constraints (as defined in Section 3.4). Corresponding tables for the Nemenyi test which include the average ranks and values for each method can be found in Appendix A.1.

`EPS` is overall much stronger than standalone `PS`. Although statistical significance does not apply between tables, we see that in terms of ACCURACY `EPS` is able to improve results by up to five percentage points on *Enron* as well as considerable gains across nearly all datasets, with the exception of *Genbase*, and also arguably *Medical*.

`EPS` also performs well against state-of-the-art methods across the various datasets and evaluation measures, as demonstrated by many statistically significant results across the tables. `EPS` clearly dominates under the EXACT-MATCH measure. This illustrates a strong advantage of `EPS`: it learns label correlations directly—i.e. by label set—as they are observed in the training set. This is also done to a lesser extent by $\texttt{RAkEL}_1$, and this method also

does relatively well under EXACT-MATCH. On the other hand, these label-set based methods can do relatively more poorly on label-based evaluation measures—like F1-MACRO$^{\times L}$—where RAkEL performs well. However, taking into account the gains on other datasets, and the margins of those gains, EPS is the dominant method overall.

The base method BR is easily outperformed by the other methods, especially on *Enron*, *Reuters*, and *Scene*, where $\texttt{EPS}_2$ achieved up to nearly 20 percentage points more under ACCURACY and AU($\overline{\text{PRC}}$). Both configurations of EPS perform well, $\texttt{EPS}_1$ particularly under ACCURACY, and $\texttt{EPS}_2$ particularly under AU($\overline{\text{PRC}}$). Both MLkNN and CLR perform particularly well under LOG-LOSS.

Note that it is not unusual that methods can differ in terms of statistical significance even though the values shown in the table are identical. More significant figures would reveal a difference. The standand deviations are very low, hence they are omitted from the table (as many significant figures would be required to display them).

BR is the fastest method, which we expect due to the simplicity of its binary models. $\texttt{EPS}_1$ is on average the fastest of the LC-based methods, even compared with simple LC (which we see when comparing between Tables 6.3 and 6.13). CLR performs relatively fast where L is small, but proves intractable otherwise. MLkNN performs very fast, however, this is not a fair comparison since MLkNN is a kNN-adapted method, whereas the other methods use SMO as their base classifier. By using more efficient base classifiers (e.g. a kNN classifier in this context), problem transformation methods like EPS can perform faster. However, we used SMO for its predictive power in this

context—which is reflected in the results.

6.9.4 Experiments on large datasets

Finally, we ran the methods on large datasets. The methods were run in a train/test scenario, and problem transformation methods employed `J48` as the base classifier (again this does not affect `MLkNN`). The change in base classifier allowed more competing methods to scale to the dimensions of more datasets and, also shows that, as generally applicable methods, the gains of our methods are not restricted to a certain classification paradigm.

Did not finish (DNF) is represented by a missing result. Results for predictive performance are displayed in Table 6.14, although results are too sparse (caused by many DNFs) to learn much from a Nemenyi test.

The high performance of `EPS` on smaller datasets is mirrored in the results on large datasets, particularly under AU($\overline{\text{PRC}}$), where $\texttt{EPS}_2$ is often several percentage points clear of its nearest competitor. `EPS` again performs particularly well under the EXACT-MATCH measure due to its ability to learn complete label sets. `EPS` gets the most number of wins over the experiment. Only on F1-MACRO$^{\times L}$—an evaluation measure which favours label-based methods like `BR`—does it not perform notably strongly.

$\texttt{RAkEL}_2$ performs relatively better on some of the larger datasets, as does `BR`. This hints that modelling entire label sets directly is not as much of an advantage where very large numbers of training examples are available: $\texttt{RAKEL}_2$ models smaller label combinations than $\texttt{RAKEL}_1$, and the trend of its better predictive performance was not apparent on smaller datasets.

`EPS` fails to perform well only on *Delicious* on account of this dataset

having too many unique label sets (indicated by its very high PUniq($\mathcal{D}$) value in Table 2.1). Pruning all these sets reduces the dataset to only a handful of training examples, as indicated by the unusually fast training time (when other methods fail to even finish). Although *Delicious* is more related to tagging than a typical multi-label problem, it shows a situation where EPS's pruning and subsambling breaks down. We also see that CLR achieves unusually low predictive performance on *MediaMill*.

Large datasets push the scalability limits of several of the algorithms. Only BR and MLkNN are able to complete on all datasets, although recall that the latter is not a fair comparison because of its different base scheme. EPS fails to scale all the way to the extremes of the *IMDB*, and EPS_2 also fails on *MediaMill*. We note that these datasets did not exist when we first introduced PS in (Read, 2008), and only more recently have datasets of this scale been commonly used for evaluations. However, we note that generally both versions of EPS are faster than RAkEL for the same datasets, and EPS_1 results in fewer DNFs than any other comparable method except BR (other methods ran out of available memory). In particular on *TMC2007*, EPS is several times faster than its closest competitor.

6.10 Conclusions and Future Work

In this chapter we have introduced and empirically demonstrated the effectiveness of the PS-paradigm. The main contributions were clearly demonstrated: 1) PS considerably reduces the running time of the standard LC method; and 2) as an ensemble framework, EPS was able to outperform a

variety of state-of-the-art methods from the literature in terms of overall predictive performance.

Although EPS's running times were similar to RAkEL (specifically RAkEL_2), we note that EPS can continue scaling to faster running times by pruning more at the base models. RAkEL_2 (where $k = 3$) represents close to the limit of scalability (which would be $k = 2$), yet already in terms of predictive performance, RAkEL_2 is not as powerful either RAkEL_1 (where $k = \frac{L}{2}$) or EPS on several datasets (i.e. *Scene*, *Slashdot* and *Yeast*). Given this gap in predictive performance, it could be argued that EPS could afford to select higher values of p and still remain competitive. Figures 6.4 and 6.5 provide an indication of potential reductions in running time relative to predictive performance. It is also notable that EPS is able to achieve its relatively low running time while still modelling complete label sets.

6.10.1 Limitations and future work.

We knew that PS's running time is never worse than LC and we saw that in practice it is much better. However, its performance is highly dependent on the underlying labelling scheme of the dataset. For atypical datasets where labelling displays very little regularity EPS ran into problems, such as on *Delicious*, where it obtained poor ACCURACY, and *IMDB* where it was unable to complete under the configured parameters. Both PS's parameters affect both efficiency and predictive performance, and the optimum values for these parameters are also dataset-dependent. In the following chapter we address these limitations with a new parameterless method based on the BR paradigm.

PS is a general method for multi-label classification as much as it is a specific multi-label classifier, and can be incorporated easily into other methods. EPS was just one example of this. RAkEL's ensemble wrapper around LC could already to scale better than LC itself, and we expect that it could scale even higher if PS was used in this ensemble instead, to combine the scalability advantages of both methods, although we leave this experimentation for future work.

Table 6.2: PS methods vs. LC: Predictive performance (with SMO).

ACCURACY

Dataset	LC	PS_1	PS_2^t	PS_2	PS_1^t
Music	0.463	0.469	0.515 ⊕	0.497	0.519 ⊕
Scene	0.717	0.720 ⊕	0.675 •	0.720 ⊕	0.674 •
Yeast	0.525	0.527	0.504 •	0.527	0.506 •
Genbase	0.971	0.966 •	0.943 •	0.958 •	0.963
Medical	0.745	0.754 ⊕	0.702 •	0.754 ⊕	0.700 •
Slashdot	0.500	0.500	0.379 •	0.502	0.357 •
20ng	0.677	0.676	0.600 •	0.676	0.602 •
LangLog	0.171	0.144 •	0.132 •	0.120 •	0.124 •
Enron	0.412	0.403 •	0.387 •	0.390 •	0.390 •
Reuters	0.490	0.497 ⊕	0.422 •	0.497 ⊕	0.404 •

⊕, • statistically significant improvement or degradation vs. LC

HAMMING-LOSS

Dataset	LC	PS_1	PS_2^t	PS_2	PS_1^t
Music	0.237	0.234	0.228	0.223	0.226
Scene	0.096	0.095 ⊕	0.107 •	0.095 ⊕	0.107 •
Yeast	0.210	0.209	0.224 •	0.209 ⊕	0.222 •
Genbase	0.003	0.004	0.007 •	0.005 •	0.005
Medical	0.013	0.012 ⊕	0.015 •	0.012 ⊕	0.015 •
Slashdot	0.049	0.049	0.060 •	0.049	0.063 •
20ng	0.033	0.033	0.038 •	0.033	0.038 •
LangLog	0.022	0.018 ⊕	0.026 •	0.017 ⊕	0.026 •
Enron	0.057	0.057 ⊕	0.066 •	0.058	0.064 •
Reuters	0.013	0.013 ⊕	0.015 •	0.012 ⊕	0.015 •

⊕, • statistically significant improvement or degradation vs. LC

Table 6.3: PS methods vs. LC: Time performance (with SMO).

Running Time (seconds)

Dataset	LC	PS_1	PS_2^t	PS_2	PS_1^t
Music	2.35	1.63 •	1.21 •	0.96 •	1.86 •
Scene	1.72	1.56 •	2.53 ⊕	1.48 •	2.63 ⊕
Yeast	89.34	31.67 •	15.01 •	12.48 •	35.65 •
Genbase	2.82	1.43 •	1.07 •	0.92 •	1.59 •
Medical	19.25	5.94 •	2.72 •	2.48 •	6.36 •
Slashdot	59.67	23.15 •	12.89 •	10.89 •	25.54 •
20ng	65.24	60.13 •	71.21 ⊕	57.29 •	73.76 ⊕
LangLog	149.25	20.28 •	7.72 •	7.00 •	21.69 •
Enron	746.73	36.89 •	5.15 •	4.81 •	38.61 •
Reuters	1058.39	148.71 •	55.59 •	43.56 •	168.52 •

⊕, • statistically significant improvement or degradation vs. LC

Table 6.4: PS methods vs. LC: Predictive performance (with J48).

ACCURACY

Dataset	LC	PS_1	PS_2^t	PS_2	PS_1^t
Music	0.424	0.400	0.427	0.423	0.423
Scene	0.572	0.576	0.575	0.577 ⊕	0.573
Yeast	0.395	0.413 ⊕	0.422 ⊕	0.415 ⊕	0.417 ⊕
Genbase	0.963	0.964	0.947 •	0.956 •	0.964
Medical	0.716	0.728	0.722	0.726	0.723
Slashdot	0.430	0.435	0.433	0.440	0.423
20ng	0.601	0.602	0.595 •	0.602	0.594 •
LangLog	0.091	0.112 ⊕	0.130 ⊕	0.107 ⊕	0.112 ⊕
Enron	0.320	0.347 ⊕	0.366 ⊕	0.353 ⊕	0.360 ⊕
Reuters	0.389	0.405 ⊕	0.410 ⊕	0.413 ⊕	0.397 ⊕

⊕, • statistically significant improvement or degradation vs. LC

HAMMING-LOSS

Dataset	LC	PS_1	PS_2^t	PS_2	PS_1^t
Music	0.280	0.288	0.280	0.279	0.278
Scene	0.150	0.148	0.146 ⊕	0.147 ⊕	0.146
Yeast	0.286	0.275 ⊕	0.269 ⊕	0.272 ⊕	0.271 ⊕
Genbase	0.004	0.004	0.005 •	0.005	0.004
Medical	0.014	0.014	0.014	0.013 ⊕	0.014
Slashdot	0.058	0.058	0.056	0.056 ⊕	0.055
20ng	0.041	0.041	0.041 •	0.040	0.041 •
LangLog	0.027	0.023 ⊕	0.027 ⊕	0.021 ⊕	0.027 ⊕
Enron	0.075	0.066 ⊕	0.069 ⊕	0.064 ⊕	0.069 ⊕
Reuters	0.018	0.016 ⊕	0.016 ⊕	0.015 ⊕	0.016 ⊕

⊕, • statistically significant improvement or degradation vs. LC

Table 6.5: EPS vs. EPS-Bagging: ACCURACY (with SMO).

Dataset	EPS-Bag.	EPS
Music	0.56±0.02	0.57±0.01
Scene	0.73±0.01	0.73±0.01
Yeast	0.54±0.01	0.55±0.01 ⊕
Genbase	0.96±0.01	0.97±0.01 ⊕
Medical	0.75±0.01	0.75±0.01
Slashdot	0.51±0.01	0.51±0.01
20ng	0.67±0.00	0.68±0.00 ⊕
LangLog	0.16±0.01	0.17±0.01 ⊕
Enron	0.44±0.01	0.45±0.01 ⊕
Reuters	0.50±0.01	0.50±0.01 •

⊕, • statistically significant improvement or degradation vs. EPS-Bagging

Table 6.6: EPS vs. EPS-Bagging: EXACT-MATCH (with SMO).

Dataset	EPS-Bag.	EPS
Music	0.30±0.02	0.32±0.02
Scene	0.68±0.02	0.69±0.02
Yeast	0.23±0.01	0.24±0.01 ⊕
Genbase	0.92±0.02	0.94±0.01 ⊕
Medical	0.66±0.02	0.65±0.02
Slashdot	0.43±0.01	0.43±0.01
20ng	0.66±0.00	0.66±0.00 ⊕
LangLog	0.23±0.01	0.25±0.01 ⊕
Enron	0.13±0.01	0.14±0.01
Reuters	0.39±0.01	0.38±0.01

⊕, • statistically significant improvement or degradation vs. EPS-Bagging

Table 6.7: EPS vs. EPS-Bagging: Running time.

Dataset	EPS-Bag.	EPS
Music	85.34± 9.29	10.30± 1.64 •
Scene	13.14± 2.80	8.41± 0.64 •
Yeast	175.72±173.21	86.02± 4.45 •
Genbase	12.30± 12.03	8.56± 0.89 •
Medical	35.53± 35.15	19.16± 2.77 •
Slashdot	102.47±179.84	53.84± 5.18 •
20ng	11.29± 17.62	7.02± 6.91 •
LangLog	97.09± 60.63	47.20± 3.67 •
Enron	155.28± 58.53	35.13± 5.69 •
Reuters	537.10±754.99	223.43±34.79 •

⊕, • statistically significantly slower or faster vs. EPS-Bagging

Table 6.8: $\texttt{EPS}_2$ vs. state-of-the-art methods: Accuracy.

Dataset	$\texttt{EPS}_2$	$\texttt{EPS}_1$	$\texttt{RAk}_1$	BR	$\texttt{RAk}_2$	CLR	MLkNN
Music	0.57	0.57	0.58	0.50 •	0.58	0.54 •	0.40 •
Scene	0.74	0.73	0.72 •	0.58 •	0.72 •	0.71 •	0.71 •
Yeast	0.55	0.55	0.55	0.50 •	0.54 •	0.53 •	0.54 •
Genbase	0.95	0.97 ⊕	0.98 ⊕	0.98 ⊕	0.98 ⊕		0.95
Medical	0.75	0.75	0.76 ⊕	0.73 •	0.75		0.62 •
Slashdot	0.52	0.51 •	0.50 •	0.43 •	0.47 •		0.30 •
20ng	0.69	0.68 •	0.69 •	0.58 •	0.65 •	0.65 •	0.37 •
LangLog	0.18	0.17 •	0.16 •	0.11 •	0.12 •		0.12 •
Enron	0.44	0.45 ⊕	0.46 ⊕	0.39 •	0.42 •		0.35 •
Reuters	0.50	0.50 •	0.45 •	0.32 •	0.30 •		0.44 •

⊕, • statistically significant improvement or degradation

Table 6.9: $\texttt{EPS}_2$ vs. state-of-the-art methods: EXACT-MATCH.

Dataset	$\texttt{EPS}_2$	$\texttt{EPS}_1$	$\texttt{RAk}_1$	BR	$\texttt{RAk}_2$	CLR	MLkNN
Music	0.32	0.32	0.31	0.26 •	0.31	0.27 •	0.13 •
Scene	0.68	0.69 ⊕	0.66 •	0.51 •	0.67 •	0.66 •	0.64 •
Yeast	0.23	0.24 ⊕	0.22 •	0.15 •	0.17 •	0.17 •	0.18 •
Genbase	0.90	0.94 ⊕	0.97 ⊕	0.97 ⊕	0.97 ⊕		0.90
Medical	0.64	0.65 ⊕	0.66 ⊕	0.65	0.65		0.50 •
Slashdot	0.43	0.43	0.41 •	0.34 •	0.36 •		0.24 •
20ng	0.66	0.66 •	0.66 •	0.50 •	0.59 •	0.60 •	0.30 •
LangLog	0.26	0.25 •	0.24 •	0.22 •	0.22 •		0.19 •
Enron	0.14	0.14	0.14	0.11 •	0.11 •		0.01 •
Reuters	0.38	0.38	0.36 •	0.27 •	0.25 •		0.29 •

⊕, • statistically significant improvement or degradation

Table 6.10: $\texttt{EPS}_2$ vs. state-of-the-art methods: AU($\overline{\text{PRC}}$).

Dataset	$\texttt{EPS}_2$	$\texttt{EPS}_1$	$\texttt{RAk}_1$	BR	$\texttt{RAk}_2$	CLR	MLkNN
Music	0.68	0.67 •	0.66 •	0.60 •	0.66 •	0.68	0.53 •
Scene	0.81	0.78 •	0.72 •	0.62 •	0.72 •	0.76 •	0.81
Yeast	0.65	0.64 •	0.63 •	0.58 •	0.63 •	0.66	0.69 ⊕
Genbase	0.95	0.96 ⊕	0.98 ⊕	0.98 ⊕	0.98 ⊕		0.97 ⊕
Medical	0.76	0.75 •	0.74 •	0.70 •	0.73 •		0.66 •
Slashdot	0.54	0.50 •	0.42 •	0.37 •	0.40 •		0.27 •
20ng	0.77	0.73 •	0.65 •	0.56 •	0.62 •	0.62 •	0.44 •
LangLog	0.12	0.10 •	0.09 •	0.07 •	0.08 •		0.11 •
Enron	0.43	0.45 ⊕	0.46 ⊕	0.36 •	0.42 •		0.46 ⊕
Reuters	0.42	0.38 •	0.33 •	0.26 •	0.25 •		0.45 ⊕

⊕, • statistically significant improvement or degradation

Table 6.11: $\texttt{EPS}_2$ vs. state-of-the-art methods: F1-MACRO$^{\times L}$.

Dataset	$\texttt{EPS}_2$	$\texttt{EPS}_1$	$\texttt{RAk}_1$	BR	$\texttt{RAk}_2$	CLR	MLkNN
Music	0.67	0.66	0.67	0.60 •	0.67	0.62 •	0.46 •
Scene	0.76	0.76 •	0.75 •	0.68 •	0.75 •	0.74 •	0.75 •
Yeast	0.41	0.41	0.41	0.33 •	0.39 •	0.38 •	0.40 •
Genbase	0.57	0.64 ⊕	0.76 ⊕	0.77 ⊕	0.76 ⊕		0.59
Medical	0.29	0.31 ⊕	0.36 ⊕	0.35 ⊕	0.35 ⊕		0.23 •
Slashdot	0.33	0.33	0.35 ⊕	0.34	0.35 ⊕		0.16 •
20ng	0.72	0.71 •	0.70 •	0.66 •	0.69 •	0.70 •	0.43 •
LangLog	0.05	0.05	0.06 ⊕	0.05 •	0.05		0.04 •
Enron	0.14	0.16 ⊕	0.21 ⊕	0.20 ⊕	0.21 ⊕		0.10 •
Reuters	0.24	0.26 ⊕	0.28 ⊕	0.22 •	0.21		0.25 ⊕

⊕, • statistically significant improvement or degradation

Table 6.12: $\texttt{EPS}_2$ vs. state-of-the-art methods: LOG-LOSS.

Dataset	$\texttt{EPS}_2$	$\texttt{EPS}_1$	$\texttt{RAk}_1$	BR	$\texttt{RAk}_2$	CLR	MLkNN
Music	3.54	3.72 •	4.13 •	5.92 •	4.04 •	3.28 ⊕	3.74 •
Scene	1.59	1.90 •	2.59 •	3.83 •	2.48 •	2.00 •	1.38 ⊕
Yeast	11.70	12.44 •	13.32 •	16.30 •	14.01 •	10.47 ⊕	10.16 ⊕
Genbase	0.78	0.70 ⊕	0.54 ⊕	0.53 ⊕	0.53 ⊕		0.77
Medical	2.08	2.23 •	2.27 •	2.60 •	2.41 •		2.54 •
Slashdot	3.76	4.41 •	5.60 •	6.37 •	5.78 •		3.71
20ng	1.90	2.36 •	3.39 •	4.57 •	3.73 •	3.35 •	3.01 •
LangLog	7.31	7.82 •	8.57 •	8.15 •	8.22 •		5.19 ⊕
Enron	11.86	12.29 •	12.40 •	14.51 •	12.90 •		10.34 ⊕
Reuters	6.27	7.06 •	8.17 •	8.99 •	9.12 •		4.57 ⊕

⊕, • statistically significant improvement or degradation

Table 6.13: EPS vs. state-of-the-art methods: Running time (seconds).

Dataset	BR	EPS_1	RAk_1	EPS_2	RAk_2	CLR	MLkNN
Music	0	13	3	42	2	0	0
Scene	6	12	14	46	14	5	3
Yeast	6	232	99	461	24	12	1
Genbase	2	11	14	44	11		2
Medical	3	36	32	87	13		0
Slashdot	17	129	83	338	95		1
20ng	507	257	1097	1230	3371	854	98
LangLog	17	97	255	222	124		3
Enron	21	164	1004	226	163		2
Reuters	22	780	1210	1604	190		4

Table 6.14: Large datasets: Predictive performance (with J48).

ACCURACY and per-dataset (rank) across methods

Dataset	BR	CLR	EPS$_1$	EPS$_2$	MLkNN	RAkEL$_1$	RAkEL$_2$
TMC2007	0.468 (5)	0.453 (6)	0.477 (4)	0.492 (2)	0.453 (6)	0.482 (3)	**0.495** (1)
Ohsumed	0.410 (6)	0.424 (3)	0.420 (4)	**0.449** (1)	0.221 (7)	0.417 (5)	0.425 (2)
IMDB	0.219 (2)				**0.230** (1)		
Bibtex	**0.319** (1)		0.300 (3)	0.317 (2)	0.182 (6)	0.285 (5)	0.298 (4)
MediaMill	0.362 (5)	0.415 (3)	**0.426** (1)		0.424 (2)		0.412 (4)
Delicious	0.188 (2)		0.003 (4)	0.024 (3)	**0.203** (1)		

EXACT-MATCH and per-dataset (rank) across methods

Dataset	BR	CLR	EPS$_1$	EPS$_2$	MLkNN	RAkEL$_1$	RAkEL$_2$
TMC2007	0.181 (5)	0.164 (7)	0.193 (3)	**0.207** (1)	0.176 (6)	0.189 (4)	0.202 (2)
Ohsumed	0.196 (3)	0.181 (6)	0.197 (2)	**0.217** (1)	0.056 (7)	0.195 (4)	0.184 (5)
IMDB	0.045 (2)				**0.051** (1)		
Bibtex	0.116 (3)		**0.136** (1)	0.133 (2)	0.042 (6)	0.090 (5)	0.101 (4)
MediaMill	0.028 (4)	0.014 (5)	**0.071** (1)		0.061 (2)		0.053 (3)
Delicious	0.002 (3)		**0.004** (1)	0.003 (2)	0.001 (4)		

AU($\overline{\text{PRC}}$) and per-dataset (rank) across methods

Dataset	BR	CLR	EPS$_1$	EPS$_2$	MLkNN	RAkEL$_1$	RAkEL$_2$
TMC2007	0.503 (6)	0.463 (7)	0.567 (2)	**0.591** (1)	0.553 (4)	0.550 (5)	0.555 (3)
Ohsumed	0.406 (6)	0.420 (5)	0.481 (2)	**0.534** (1)	0.263 (7)	0.433 (3)	0.428 (4)
IMDB	0.217 (2)				**0.248** (1)		
Bibtex	0.290 (3)		0.293 (2)	**0.338** (1)	0.225 (6)	0.286 (4)	0.262 (5)
MediaMill	0.374 (4)	0.044 (5)	0.548 (2)		**0.550** (1)		0.510 (3)
Delicious	**0.125** (1)		0.023 (3)	0.023 (3)	0.124 (2)		

F1-MACRO$^{\times L}$ and per-dataset (rank) across methods

Dataset	BR	CLR	EPS$_1$	EPS$_2$	MLkNN	RAkEL$_1$	RAkEL$_2$
TMC2007	0.517 (3)	0.455 (6)	0.489 (5)	0.499 (4)	0.425 (7)	0.518 (2)	**0.530** (1)
Ohsumed	0.398 (3)	**0.408** (1)	0.376 (6)	0.399 (2)	0.097 (7)	0.380 (5)	0.390 (4)
IMDB	**0.085** (1)				0.045 (2)		
Bibtex	0.287 (2)		0.188 (5)	0.216 (4)	0.152 (6)	0.244 (3)	**0.297** (1)
MediaMill	0.146 (4)	0.102 (5)	0.149 (3)		0.153 (2)		**0.163** (1)
Delicious	0.085 (2)		0.004 (4)	0.014 (3)	**0.099** (1)		

LOG-LOSS and per-dataset (rank) across methods

Dataset	BR	CLR	EPS$_1$	EPS$_2$	MLkNN	RAkEL$_1$	RAkEL$_2$
TMC2007	5.8 (5)	5.8 (5)	5.2 (3)	**4.4** (1)	4.6 (2)	5.6 (4)	5.8 (5)
Ohsumed	5.7 (4)	5.3 (6)	5.9 (3)	**4.6** (1)	5.4 (5)	7.0 (2)	7.1 (6)
IMDB	7.5 (2)				**6.5** (1)		
Bibtex	**10.0** (1)		13.1 (5)	11.0 (3)	10.8 (2)	12.3 (4)	13.5 (6)
MediaMill	18.9 (5)	18.7 (4)	16.7 (2)		**13.9** (1)		18.3 (3)
Delicious	109.7 (2)		152.2 (3)	157.2 (4)	**105.7** (1)		

Chapter 7
Classifier Chains

In (Read et al., 2009b) we introduced *classifier chains* for multi-label classification. The primary goal of this method is to overcome the issues of BR, while retaining its advantages. An ensemble version of classifier chains is able to compete with state-of-the-art methods and scale to large datasets. This chapter presents classifier chains, additionally details improvements to the ensemble process, and carries out a more extensive experimental evaluation.

7.1 In Defence of Binary Methods

The binary relevance (BR) approach is widely discarded in the literature on the basis of its label independence assumption. There is good reason for this. Most methods in the literature use BR as a baseline method and are easily able to improve its predictive performance. However, let us review the advantages of BR, which are rarely discussed.

BR is theoretically simple, and resistant to combination-overfitting, since it does not expect examples to be associated with previously-observed combinations of labels (as in LC methods), and can therefore easily handle irregular

labelling. Since labels have a one-to-one relationship with binary models, labels can in theory be added and removed without affecting the rest of the model.

One of the most important advantages of BR is its computational complexity as compared with other methods. Given a constant number of examples BR scales linearly with the number of labels, whereas PW methods require a number of models quadratic to the number of labels and, although LC and RT-based methods involve only a single model, this model must learn multiple class labels.

We next present the classifier chains method, which overcomes the label independence assumption of BR while maintaining many of BR's favourable aspects, including its low computational complexity.

7.2 The Classifier Chains Method (CC)

The *classifier chain* model (CC) involves L binary transformations—one for each label—as in BR. CC is different from BR in that the attribute space for each binary model is extended with the 0/1 label relevances of all previous classifiers; thus forming a *classifier chain*. The training procedure is outlined in Algorithm 7.1. Figure 7.1 illustrates the process with an example, contrasting it with that of BR.

Hence a chain $\boldsymbol{h} = (h_1, \ldots, h_L)$ of binary classifiers is formed. Each classifier h_j in the chain is responsible for learning and predicting the binary association of the jth label given the attribute space, augmented by all prior binary relevance predictions in the chain. Note that, although the original

Algorithm 7.1 `CC`'s training phase.

```
TRAINING(D = {(x_1, y_1), ..., (x_N, y_N)})
 1  for j = 1, ..., L
 2       do ▷ the jth binary transformation and training
 3          D'_j ← {}
 4          for (x, y) ∈ D
 5               do x' ← [x_1, ..., x_P, y_1, ..., y_{j-1}]
 6                  D'_j ← D'_j ∪ (x', y_j)
 7          ▷ train h_j to predict binary relevance of y_j
 8          h_j : D'_j → {0, 1}
 9  ▷ return a classifier chain
10  return (h_1, ..., h_L)
```

instance $\boldsymbol{x}$ in Figure 7.1 has binary attributes, instances may also be composed of other types of attributes such as numerical values. The chained attributes, however, are always binary.

It is straightforward to obtain classifications from this chain. The classification process begins at the first classifier h_1 and propagates predictions along the chain: the jth binary classifier predicts the relevance of the jth label, given the attribute space augmented by the predictions of all previous binary classifiers in the chain. This prediction process is outlined in Algorithm 7.2.

This chaining method passes label information efficiently between classifiers, allowing `CC` to learn label correlation information and thus overcoming the problem of `BR` that label correlations cannot be explicitly taken into account at prediction time.

The order of the chain itself (which is determined by the order of the label

Figure 7.1: Transformation under BR and CC for $(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{y} = [1, 0, 0, 1, 0]$ and $\boldsymbol{x} = [0, 1, 0, 1, 0, 0, 1, 1, 0]$ (assuming, for simplicity, a binary attribute space). Each classifier h_j is trained to predict $\boldsymbol{y}_j \in \{0, 1\}$.

(a) BR's transformation

$\boldsymbol{h}:$	$\boldsymbol{x} \rightarrow$	$\boldsymbol{y}$
h_1:	[0,1,0,1,0,0,1,1,0]	1
h_2:	[0,1,0,1,0,0,1,1,0]	0
h_3:	[0,1,0,1,0,0,1,1,0]	0
h_4:	[0,1,0,1,0,0,1,1,0]	1
h_5:	[0,1,0,1,0,0,1,1,0]	0

(b) CC's transformation

$\boldsymbol{h}:$	$\boldsymbol{x}' \rightarrow$	$\boldsymbol{y}$
h_1:	[0,1,0,1,0,0,1,1,0]	1
h_2:	[0,1,0,1,0,0,1,1,0,1]	0
h_3:	[0,1,0,1,0,0,1,1,0,1,0]	0
h_4:	[0,1,0,1,0,0,1,1,0,1,0,0]	1
h_5:	[0,1,0,1,0,0,1,1,0,1,0,0,1]	0

Algorithm 7.2 CC's prediction phase for a test instance $\boldsymbol{x}$.

CLASSIFY($\boldsymbol{x}$)
1 ▷ global $(h_1, \ldots, h_L)$
2 $\hat{\boldsymbol{y}} = [\hat{y}_1, \ldots, \hat{y}_L]$ ▷ a vector where each $\hat{y}_j = 0$
3 **for** $j = 1,\ldots,L$
4 **do** $\boldsymbol{x}' \leftarrow [x_1, \ldots, x_P, \hat{y}_1, \ldots, \hat{y}_{j-1}]$
5 $\hat{y}_j \leftarrow h_j(\boldsymbol{x}')$
6 **return** $\hat{\boldsymbol{y}}$

indices in label-set vector representations of $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$) clearly has an effect on predictive performance. Possible heuristics for selecting a chain order for CC could be based on label correlations measured in the training data; or the performance of the binary classifiers (determined by internal validation). However, we discovered that an ensemble framework, in which a different random chain ordering is used at each iteration, is a more powerful alternative. Section 7.4 presents this framework. First let us take a closer look at the time complexity of the CC method as compared with other methods.

7.3 Time Complexity

Although an average of $L/2$ attributes is added to each instance, the size of the label space—L—is limited in practice, and therefore the computational complexity of CC can be very close to BR. BR's complexity is $O(L \times f(M, N))$, where $f(M, N)$ is the complexity of the underlying learner. CC's complexity is $O(L \times f(M + L, N))$, i.e. a penalty is incurred for having up to L additional attributes. Assuming a linear base learner (with respect to the number of attributes), CC's complexity becomes $O(L \times M \times N + L \times L \times N)$, where the first term dominates as long as $L < M$, which we already expect (see the assumptions outlined in Chapter 1), and therefore the effective complexity of CC is $O(L \times M \times N)$, which is identical to BR's complexity.

We can see how this translates into practice: Figure 7.2 shows how different methods scale with respect to the size of the label space (L). For this plot, we have generated artificial datasets to vary the size of L in a controlled fashion with a constant $N = 5000$, $M = 500$ (numeric attributes), and $\text{LCard}(\mathcal{D}) = 1.0 + n0.15$ where $L = 2^n$ (a rough approximation suitable for measuring time complexity). For this purpose, we used the framework for generating synthetic multi-label data that we introduced in (Read et al., 2009a). For Figure 7.3 the same analysis is carried out, but varying instead the number of examples N with a constant $L = 10$ and $M = 20$ (again, numeric attributes). Note that, although we have often used N to refer to the number of training examples, here N is the number of training *and* testing examples, in a 60 : 40 ratio. A selection of algorithms were run on these datasets to analyse their running time with respect to L and N. SMO is the base learner for all problem transformation methods (this excludes IBLR

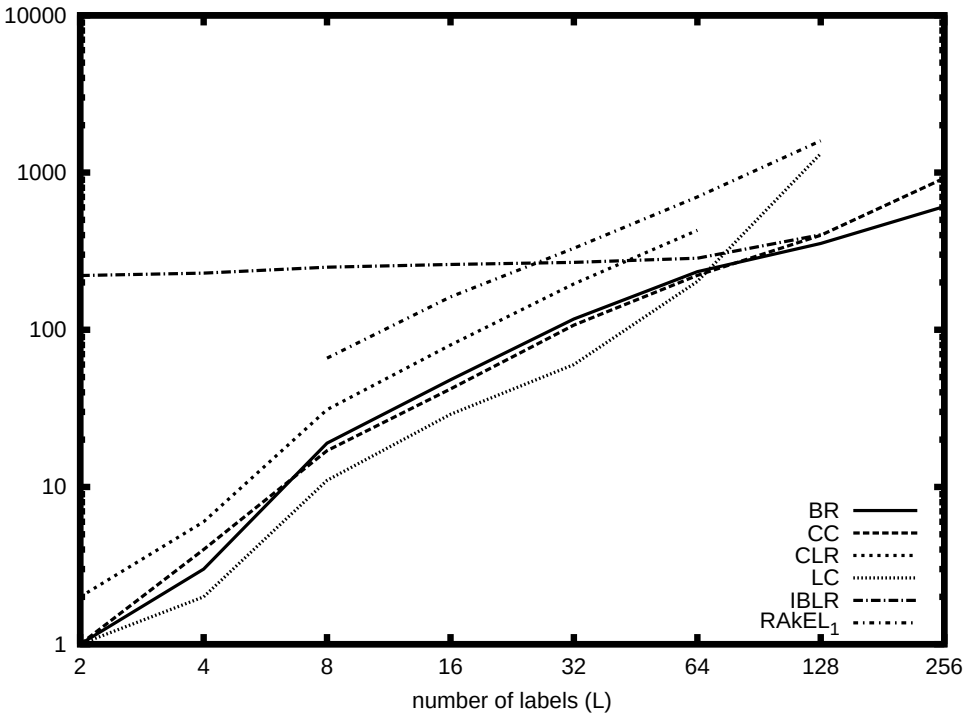


Figure 7.2: Running times for methods on artificial datasets with $L = 2, 4, \ldots, 256$ (note the logarithmic scale).

which is an adaptation of kNN and logistic regression).

We emphasise that some lines end prematurely in both plots, which indicates that algorithms were unable to complete due to lack of memory (we allowed 1GB).

With respect to the number of labels (L), the binary method `BR` scales approximately linearly, and `CC` only diverges from `BR` when $L > 128$. As expected, `CLR` is very sensitive to L with respect to memory. It becomes intractable when $L > 64$ (at $L = 128$ this method would need 16256 models). `LC` is able to complete up to $L = 128$, at which point its greater-than-linear complexity with respect to L is already clear. `RAkEL` appears to scale about

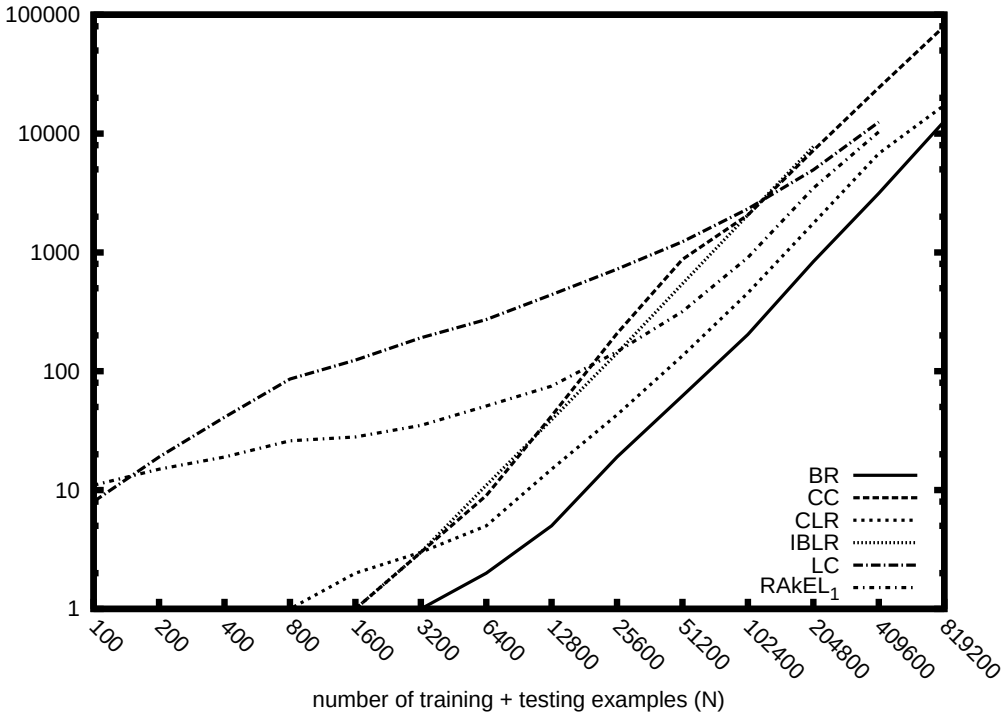


Figure 7.3: Running times for methods on artificial datasets with $N = 100, 200, \ldots, 819200$ (note the logarithmic scale).

linearly, but the memory it uses also prevent it from finishing when $L = 256$, and having multiple models makes it take even longer than LC in this context. IBLR's running times are initially almost constant with respect to L (which is plausible: counting the number of labels in a neighborhood can be a very efficient operation) but memory requirements obviously become high, since this method does not finish when $L = 256$.

With respect to the number of examples (N), LC and RAkEL both scale much higher than other methods for lower numbers of examples, and neither complete when $N = 819200$. As expected, the kNN-based IBLR scales worst for large N, since kNN is N-sensitive; not completing when $N > 204800$. Under our assumptions (see Section 1.1) L is inherently limited in scope and not expected to grow without bound. This does not apply to N, and, as N becomes larger, kNN methods will reach their natural limits of scalability. Only CC, BR, and CLR (which needs in this case only $L(L-1)/2 = 10*9/2 = 45$ classifiers) completes when $N = 819200$. Under SMO, CC's running time is progressively affected by the additional attributes in terms of time, but it is the only method that explicitly models label correlations to complete for the full range of N.

Overall, only the binary methods BR and CC scale to the extremes in both dataset dimensions. IBLR scales well with respect to L in terms of running time, but not with respect to N and vice versa for CLR. For growing N, RAkEL is able to improve its running time over LC, but neither can complete on the largest numbers of labels and examples.

7.4 Ensembles of Classifier Chains (ECC)

Chapter 6 discussed the benefits of using ensembles and provided positive results with respect to the EPS method. We now present *ensembles of classifier chains* (ECC). Unlike EPS, instances are not already being duplicated within the multi-label models, and therefore a standard bagging scheme is appropriate. Thus, in this chapter we obtain better results than in (Read et al., 2009b) where we used a simple-subset ensemble.

ECC trains m CC classifiers using a standard bagging scheme, where the binary models of each chain are ordered according to a random seed. Each model is then likely to be unique and can predict different label sets from other models. The label sets are combined across models as votes into a final label-set prediction with the classification scheme outlined in Algorithm 6.6.

We also employ BR under this generic ensemble framework to create an *ensemble of binary relevance classifiers* (EBR). As far as we are aware, such a scheme had not been previously evaluated in the literature until the work in (Read et al., 2009b). Comparing with EBR empirically will help to evaluate and analyse the performance of ECC.

With respect to computational complexity, ECC is $O(m \times \texttt{CC})$ for m iterations. Section 7.7 on future work discusses how this complexity can be reduced.

7.5 Related Work

Chapter 5 reviewed work related to CC, most notably the MBR and SMBR methods.

MBR uses BR's binary outputs as additional attributes in a meta classifier to take into account label correlations in a BR-scheme. IBLR can be seen as a specialisation of the MBR approach using kNN with logistic regression as the meta method. CC also uses binary outputs as additional attributes. However, CC does not require the additional meta classifier and its associated complexity (i.e. the extra iterations of the data at training and testing time for the meta classifier).

SMBR involves a faster meta process, which maps BR's binary outputs directly to label sets from the training set, and therefore does not require internal classification at training time. However, this method is not as adaptable as CC, since it can only fit label sets which were observed in the training data. In this sense, SMBR suffers some of the disadvantages of LC.

7.6 Experiments

7.6.1 CC against BR and related methods.

Initially we compare standalone CC to BR and BR-related methods MBR and SMBR, which we reproduce in our framework.

Results of a 5×2 cross validation experiment are shown in Table 7.1 under a corrected paired t-test against CC on various evaluation measures. A selection of running times are plotted in Figure 7.4. SMO is used as the base classifier. These results show the value of CC's chaining method. Overall, CC improves convincingly over both the default BR method and related methods MBR and SMBR, with the exception of the small *Music* dataset. Most performance gains are statistically significant and the running times support the

theory presented in Section 7.3: BR is naturally the fastest and in practice the complexity added to BR by CC is not noticeable except when L is relatively large such as in *Reuters*. In some cases, CC or SMBR are even faster than BR, although this can be attributed to minor variations in implementation and run-time conditions. MBR's two-stage stacking process means that its running times are roughly twice that of BR.

7.6.2 ECC against state-of-the-art methods.

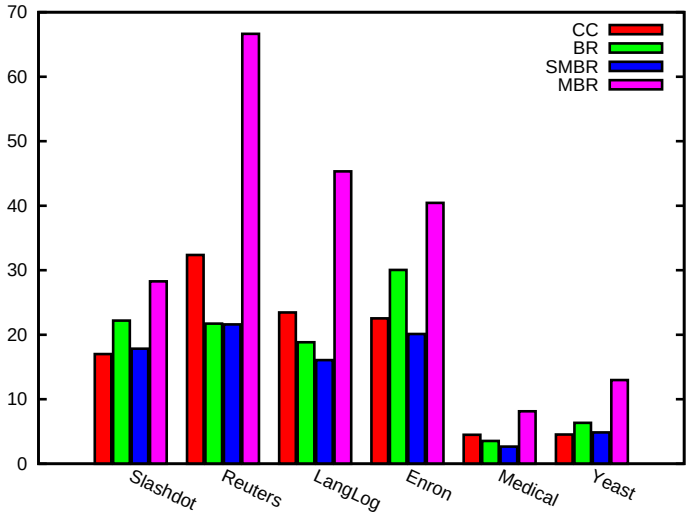
As the main focus of our evaluation, we compare the ensemble method ECC to EBR and various state-of-the-art algorithms. MLkNN has arguably been superceded recently by the newer IBLR (which is also of interest since it is closely related to MBR), so we compare with this method instead. We also compare with CLR, RAkEL, and our own EPS method from Chapter 6. The parameter variations for the latter two methods are described in the evaluation section of that chapter. We also create two variations of both EBR and ECC: $\texttt{EBR}_1$ and $\texttt{ECC}_1$ carry out 10 iterations, and $\texttt{EBR}_2$ and $\texttt{ECC}_2$ carry out 50 iterations.

Again we perform 5×2 cross validation and record significance under a corrected paired t-test with SMO as the base classifier for problem transformation methods (excluding kNN- and logistic regression-based IBLR). The results are displayed in Tables 7.2 (ACCURACY), 7.3 (EXACT-MATCH), 7.4 (AU($\overline{\text{PRC}}$)), 7.5 (F1-MACRO$^{\times L}$), 7.6 (LOG-LOSS), and 7.7 (RUNNING-TIME). Corresponding tables for the Nemenyi test, which include average ranks and values, can be found in Appendix A.2.

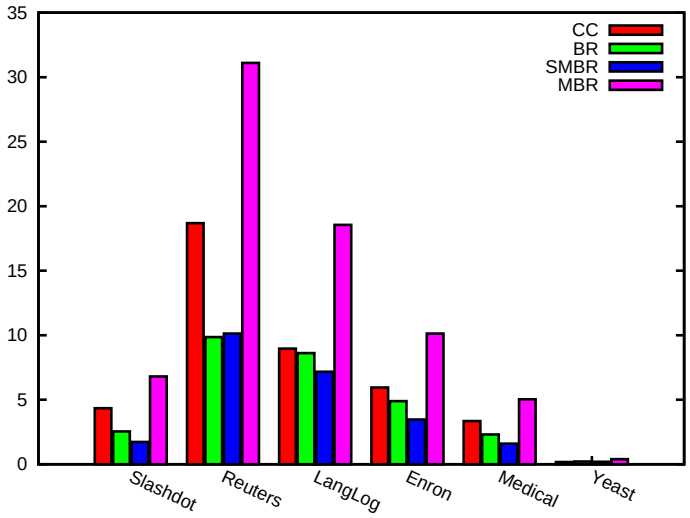
The results show that ECC is the dominant method. This is particu-

Figure 7.4: Time measurements.

(a) Training times (seconds).



(b) Testing times (seconds).



larly clear under LOG-LOSS and AU($\overline{\text{PRC}}$), where $\texttt{ECC}_2$ excels. Even though LOG-LOSS and AU($\overline{\text{PRC}}$) are label-based evaluation measures (and, as such, should favour BR-type methods like CC), ECC is not designed to optimise either of them. Furthermore, ECC performs well overall, even under label-set-based evaluation measures like accuracy. Only under EXACT-MATCH do other methods (namely $\texttt{EPS}_{1,2}$ and $\texttt{RAkEL}_1$) have a noticeable advantage because they model label sets.

ECC is thus consistent in its high performance across all the datasets, and there are no results where ECC performs poorly, unlike—for example—IBLR on *Enron*, CLR on *Music*, or $\texttt{RAk}_1$ on *LangLog*.

Generally IBLR performs well on LOG-LOSS relative to other measures. CLR performs well on several datasets but its complexity prohibits its completion on anything but small-L datasets.

The effectiveness of ECC's chaining method is further demonstrated in comparison with EBR – these methods only differ in terms of the chaining method. ECC gains a clear advantage overall, and particularly on datasets like *Reuters* and evaluation measures like EXACT-MATCH.

On the other hand, EBR (specifically $\texttt{EBR}_2$) performs surprisingly well in some areas—like on *Slashdot* and *20ng*—as compared with other methods. This is interesting, considering that this method is simply an ensemble of the baseline method BR.

Both EBR and ECC make consistent gains with respect to the number of iterations: $\texttt{EBR}_2$ and $\texttt{ECC}_2$ (50 iterations) perform significantly better than $\texttt{EBR}_1$ and $\texttt{ECC}_1$ (10 iterations), particularly under ACCURACY and AU($\overline{\text{PRC}}$).

7.6.3 Experiments on large datasets.

We also run the competing methods on large datasets, with the same setup we used in Chapter 6: a train/test scenario, where the problem transformation methods employ `J48` as the base classifier (which excludes `IBLR`).

Results for predictive performance are displayed in Table 7.8. Although too many methods *did not finish* (DNF)—shown as missing values in the table—to learn much from a statistical significance test, the ranks (across methods, for each dataset) are included. Table 7.9 displays corresponding run-time results.

Again `ECC` excels in terms of predictive performance. However, on large datasets, performance is best under the label-set-based measures ACCURACY and EXACT-MATCH, more so than label-based measures. The fact that `ECC` now does particularly well under label-set evaluation hints that, given enough training instances, variance in the labelling is reduced so much by the chain model that `ECC` begins to behave more like label-set methods (i.e. `RAkEL` and `EPS`) which model label-set combinations. Although this could be partially due to the change in base classifier, we see also on standard datasets that `ECC` was relatively better at EXACT-MATCH on larger datasets (e.g. *Reuters*) than smaller datasets (e.g. *Music*).

Thus, label-set-based methods begin to overfit large training sets in terms of label-correlations. Moreover `EBR` performs particularly well on large datasets, which also lends weight to the intuition that when high numbers of training examples are involved methods get relatively less benefit for modelling label correlations because it is possible to model individual labels so well.

Section 7.3 discussed the time complexity of various methods, which was

mostly borne out in the experiments on large datasets. IBLR runs out of memory dealing with large-L and slows down considerably under many training examples; CLR only performs where L is small; and in spite of high worst-case complexity, EPS can perform reasonably in practice on most datasets.

On several datasets, ECC's running times can be up to several times greater than other methods. This is mainly regarding large datasets of large L (e.g. *Bibtex*, *Delicious*, and *MediaMill*). However, the fact that EBR's running times are about as high on these datasets (with the exception of *Delicious*) leads us to conclude that the number of binary models is the biggest influence on these times, rather than the extra binary attributes in the attribute space of the binary transformations for the CC models. Although the competing methods have a higher theoretical computational complexity, they often perform efficiently in practice. However, this speed comes at a price: these methods often either fail to perform well (for example CLR on *MediaMill*, RAkEL on *Bibtex*, and EPS on *Delicious*), or fail to perform within memory bounds. The latter case is clear, as the binary methods are the only methods able to finish on all the large datasets.

Thus, although ECC can be time-intensive, its running time is straightforward to estimate and theoretically lower than other methods and it can deal with the entire collection of datasets under memory constraints. And, importantly, its predictive performance is correspondingly as reliable.

7.7 Limitations and Future Work

7.7.1 Achieving more efficient chains

When L becomes large relative to M or N `CC` performs notably slower than `BR`, for example on *LangLog* and *Reuters*. This is due to the extra attribute space added to `BR`, which is up to $L-1$ extra attributes (in the final transformation in the chain). Clearly, this issue is exacerbated for multiple models, i.e. `ECC`.

Further work can make the running time of `CC` much closer to `BR` without significant degradation of predictive performance. Analysis in Chapter 2 showed that domain-dependent label correlations are invariably only a small subset of all possible label correlations, and therefore a partial chain can in theory sufficiently model any dataset: each binary model of a classifier chain would only need to model a subset of the binary outputs from previous models in the chain (where correlations are significant). Correlations can be detected and measured with, for example, the *correlation coefficient* measure introduced by Tsoumakas et al. (2009a). Depending on the correlations detected, running time—especially on large-L datasets—could be improved substantially. In future work we intend to investigate this.

The results on large datasets also suggest that when the number of training examples grows very large, modelling label correlations not only becomes less effective, but may even eventually become a hinderence. Therefore, chained classifiers with fewer additional attributes in their transformed problem may even achieve higher predictive performance as well as greater efficiency.

7.7.2 Achieving more efficient ensembles

As well as the issue of chain length, ECC can face a problem with the number of models it requires in some contexts—a problem it shares with EBR. Even though each ensemble iteration only deals with simple binary models, these are complete with respect to the attribute space and training set ($\mathcal{X}$ and $\mathcal{D}$). Under m iterations, ECC creates $m \times L$ binary models, each of which sees N transformed instances. ECC_2 requires nearly $8,000$ models on the *Bibtex* dataset, and over 20 million training examples are drawn from *IMDB* over $m = 10$ iterations. The results clearly illustrated the time costs.

Existing work on BR-related problems have addressed the issue of complexity across all dimensions with a variety of strategies and, because CC is based closely on BR, it can also employ these strategies.

Ráez et al. (2004) show that the most imbalanced binary classifiers from a BR transformation can be removed entirely for a significant reduction of computational complexity without significantly hindering predictive performance. We suggest that any lost performance could be recovered (or even improved upon) by building a single-model method (e.g. PS) on these labels.

Another strategy discussed by Ráez et al. (2004) is *down-sampling*: selecting fewer negative examples for each binary model to reduce binary imbalances (to improve prediction), with the additional benefit of reduced training time.

As well as the label and example space, it is also possible to sub-sample the attribute space; an approach used by Yan et al. (2007). In this way, each binary model contains fewer than M attributes, but hopefully these attributes are the ones relevant to that particular binary problem.

Given the results reported by these methods we can expect large reductions in training time when applied to ECC.

7.8 Summary and Conclusions

The dominance of CC over BR and related methods, and that of ECC over EBR and state-of-the-art methods, showed the value of classifier chains: CC can model label correlations for higher predictive performance without overfitting the training data and without incurring significant increases in running time over BR, and ECC provides even higher predictive performance on account of being less prone to variations in the labelling scheme than other methods.

There were no cases where ECC failed to perform well and it usually performs best. Unlike other competing methods like RAkEL and EPS, ECC requires no parameter configuration other than the generic ensemble parameter of the number of iterations, which is intuitive to calibrate: more iterations provide better performance at a calculable cost of time complexity. As such, ECC can be seen as a good generally-applicable 'off the shelf' method: it is easily configurable, has a relatively low worst case time performance, and it is robust with respect to predictive performance across a variety of datasets.

Even without additional improvements to efficiency, we can say that CC and ECC are both scalable methods. CC does not suffer from the same memory limitations of single-model multi-class transformations (e.g. EPS, RAkEL) or meta methods (e.g. MBR, IBLR), and ECC successfully trained on all the datasets in our collection.

Table 7.1: CC vs. related methods: Predictive performance.

ACCURACY

Dataset	CC	BR	SMBR	MBR
Music	0.45±0.05	0.50±0.01 ⊕	0.52±0.01 ⊕	0.52±0.01 ⊕
Scene	0.67±0.01	0.58±0.01 •	0.62±0.01 •	0.61±0.01 •
Yeast	0.51±0.01	0.50±0.01 •	0.51±0.01	0.50±0.01 •
Genbase	0.98±0.01	0.98±0.01	0.97±0.02	0.98±0.01
Medical	0.76±0.01	0.73±0.01 •	0.73±0.01 •	0.73±0.01 •
Slashdot	0.46±0.01	0.43±0.01 •	0.45±0.01 •	0.44±0.01 •
20ng	0.62±0.00	0.58±0.00 •	0.59±0.00 •	0.58±0.00 •
LangLog	0.11±0.01	0.11±0.01	0.11±0.01	0.11±0.01
Enron	0.40±0.01	0.39±0.01 •	0.41±0.01 ⊕	0.39±0.01 •
Reuters	0.40±0.02	0.32±0.00 •	0.33±0.01 •	0.32±0.01 •

⊕, • statistically significant improvement or degradation vs. CC

EXACT-MATCH

Dataset	CC	BR	SMBR	MBR
Music	0.26±0.02	0.26±0.02	0.27±0.02	0.27±0.01
Scene	0.62±0.01	0.51±0.01 •	0.57±0.01 •	0.55±0.01 •
Yeast	0.20±0.01	0.15±0.01 •	0.18±0.01 •	0.15±0.01 •
Genbase	0.97±0.01	0.97±0.01	0.95±0.02 •	0.97±0.01
Medical	0.68±0.02	0.65±0.02 •	0.66±0.02 •	0.65±0.02 •
Slashdot	0.38±0.01	0.34±0.01 •	0.37±0.01 •	0.34±0.01 •
20ng	0.57±0.01	0.50±0.01 •	0.56±0.00 •	0.51±0.01 •
LangLog	0.22±0.01	0.22±0.01	0.23±0.01	0.22±0.01
Enron	0.12±0.01	0.11±0.01 •	0.14±0.01 ⊕	0.11±0.01 •
Reuters	0.34±0.01	0.27±0.01 •	0.29±0.01 •	0.27±0.01 •

⊕, • statistically significant improvement or degradation vs. CC

F1-MACRO$^{\times L}$

Dataset	CC	BR	SMBR	MBR
Music	0.26±0.02	0.26±0.02	0.27±0.02	0.27±0.01
Scene	0.62±0.01	0.51±0.01 •	0.57±0.01 •	0.55±0.01 •
Yeast	0.20±0.01	0.15±0.01 •	0.18±0.01 •	0.15±0.01 •
Genbase	0.97±0.01	0.97±0.01	0.95±0.02 •	0.97±0.01
Medical	0.68±0.02	0.65±0.02 •	0.66±0.02 •	0.65±0.02 •
Slashdot	0.38±0.01	0.34±0.01 •	0.37±0.01 •	0.34±0.01 •
20ng	0.57±0.01	0.50±0.01 •	0.56±0.00 •	0.51±0.01 •
LangLog	0.22±0.01	0.22±0.01	0.23±0.01	0.22±0.01
Enron	0.12±0.01	0.11±0.01 •	0.14±0.01 ⊕	0.11±0.01 •
Reuters	0.34±0.01	0.27±0.01 •	0.29±0.01 •	0.27±0.01 •

⊕, • statistically significant improvement or degradation vs. CC

Table 7.2: $\texttt{ECC}_2$ vs. other methods: ACCURACY.

Dataset	$\texttt{ECC}_2$	$\texttt{EPS}_1$	$\texttt{EPS}_2$	$\texttt{RAkEL}_1$	$\texttt{ECC}_1$	$\texttt{EBR}_1$	$\texttt{EBR}_2$	$\texttt{RAk}_2$	IBLR	CLR
Music	0.56	0.57	0.57⊕	0.57 ⊕	0.55•	0.55	0.55	0.58⊕	0.56	0.54•
Scene	0.71	0.73⊕	0.74⊕	0.72	0.70•	0.69•	0.71 •	0.72	0.73⊕	0.71•
Yeast	0.54	0.55⊕	0.55⊕	0.54 ⊕	0.53•	0.53•	0.53 •	0.53	0.54	0.53•
Genbase	0.98	0.97 •	0.95 •	0.98	0.98	0.98	0.98	0.98⊕	0.95 •	
Medical	0.79	0.75 •	0.75 •	0.76 •	0.78•	0.77•	0.78	0.52 •	0.54 •	
Slashdot	0.52	0.51 •	0.52 •	0.51 •	0.51•	0.51•	0.53⊕	0.47 •	0.26 •	
20ng	0.69	0.68 •	0.69	0.69 •	0.68•	0.68•	0.70⊕	0.65 •	0.39 •	0.65•
LangLog	0.17	0.17	0.18⊕	0.16 •	0.14•	0.14•	0.18	0.11 •	0.04 •	
Enron	0.47	0.45 •	0.44 •	0.46 •	0.46•	0.46•	0.47 •	0.42 •	0.34 •	
Reuters	0.49	0.50⊕	0.50⊕	0.45 •	0.46•	0.37•	0.41 •	0.30 •	0.38 •	

⊕, • statistically significant improvement or degradation vs. $\texttt{ECC}_2$

Table 7.3: $\texttt{ECC}_2$ vs. other methods: EXACT-MATCH.

Dataset	$\texttt{ECC}_2$	$\texttt{EPS}_1$	$\texttt{EPS}_2$	$\texttt{RAk}_1$	$\texttt{ECC}_1$	$\texttt{EBR}_1$	$\texttt{EBR}_2$	$\texttt{RAk}_2$	IBLR	CLR
Music	0.29	0.32⊕	0.32⊕	0.31⊕	0.27	0.26•	0.27 •	0.32⊕	0.30	0.27 •
Scene	0.65	0.69⊕	0.68⊕	0.66⊕	0.64	0.63•	0.65	0.67⊕	0.66⊕	0.66⊕
Yeast	0.19	0.24⊕	0.23⊕	0.22⊕	0.19	0.15•	0.15 •	0.16	0.19	0.17 •
Genbase	0.95	0.94	0.90 •	0.96⊕	0.95	0.95	0.95	0.97⊕	0.91 •	
Medical	0.69	0.65 •	0.64 •	0.67 •	0.68•	0.67•	0.68 •	0.45 •	0.42 •	
Slashdot	0.42	0.43⊕	0.43⊕	0.42	0.42	0.42	0.42⊕	0.36 •	0.15 •	
20ng	0.64	0.66⊕	0.66⊕	0.66⊕	0.63•	0.63•	0.65⊕	0.59 •	0.32 •	0.60 •
LangLog	0.25	0.25	0.26⊕	0.24 •	0.22•	0.22•	0.25	0.21 •	0.14 •	
Enron	0.12	0.14⊕	0.14⊕	0.14⊕	0.12	0.11•	0.12 •	0.11 •	0.02 •	
Reuters	0.34	0.38⊕	0.38⊕	0.36⊕	0.32•	0.26•	0.29 •	0.25 •	0.25 •	

⊕, • statistically significant improvement or degradation vs. $\texttt{ECC}_2$

Table 7.4: $\texttt{ECC}_2$ vs. other methods: AU($\overline{\text{PRC}}$).

Dataset	$\texttt{ECC}_2$	$\texttt{EPS}_1$	$\texttt{EPS}_2$	$\texttt{RAk}_1$	$\texttt{ECC}_1$	$\texttt{EBR}_1$	$\texttt{EBR}_2$	$\texttt{RAk}_2$	IBLR	CLR
Music	0.69	0.67•	0.68 •	0.66•	0.68•	0.67•	0.68 •	0.67•	0.69	0.68•
Scene	0.81	0.78•	0.81	0.72•	0.78•	0.73•	0.77 •	0.72•	0.82⊕	0.76•
Yeast	0.67	0.64•	0.65 •	0.63•	0.66•	0.64•	0.64 •	0.62•	0.69⊕	0.66•
Genbase	0.99	0.96•	0.95 •	0.98•	0.98	0.98	0.98	0.98	0.96 •	
Medical	0.81	0.75•	0.76 •	0.74•	0.79•	0.78•	0.80 •	0.51•	0.59 •	
Slashdot	0.57	0.50•	0.54 •	0.43•	0.52•	0.48•	0.54 •	0.40•	0.29 •	
20ng	0.79	0.73•	0.77 •	0.65•	0.75•	0.70•	0.75 •	0.62•	0.47 •	0.62•
LangLog	0.10	0.10	0.12⊕	0.09•	0.08•	0.08•	0.11⊕	0.07•	0.06 •	
Enron	0.53	0.45•	0.43 •	0.46•	0.51•	0.51•	0.52 •	0.42•	0.45 •	
Reuters	0.42	0.38•	0.42	0.33•	0.37•	0.34•	0.37 •	0.25•	0.40 •	

⊕, • statistically significant improvement or degradation vs. $\texttt{ECC}_2$

Table 7.5: $\texttt{ECC}_2$ vs. other methods: F1-MACRO$^{\times L}$.

Dataset	$\texttt{ECC}_2$	$\texttt{EPS}_1$	$\texttt{EPS}_2$	$\texttt{RAk}_1$	$\texttt{ECC}_1$	$\texttt{EBR}_1$	$\texttt{EBR}_2$	$\texttt{RAk}_2$	IBLR	CLR
Music	0.66	0.66	0.67⊕	0.67⊕	0.65•	0.64•	0.65 •	0.67⊕	0.66	0.62 •
Scene	0.75	0.76	0.76⊕	0.74	0.73•	0.73•	0.74 •	0.75	0.75	0.74 •
Yeast	0.37	0.41⊕	0.41⊕	0.41⊕	0.37	0.37	0.37	0.38	0.41⊕	0.38⊕
Genbase	0.75	0.64 •	0.57 •	0.76	0.75	0.75	0.75	0.76	0.63 •	
Medical	0.35	0.31 •	0.29 •	0.35	0.35	0.35	0.35	0.25	0.20 •	
Slashdot	0.36	0.33 •	0.33 •	0.35	0.34•	0.34•	0.36	0.35	0.15 •	
20ng	0.73	0.71 •	0.72 •	0.70 •	0.71•	0.71•	0.73 •	0.69 •	0.45 •	0.70 •
LangLog	0.06	0.05	0.05	0.06⊕	0.05	0.05	0.06	0.05	0.02 •	
Enron	0.20	0.16 •	0.14 •	0.21⊕	0.20	0.20	0.20	0.21⊕	0.13 •	
Reuters	0.27	0.26 •	0.24 •	0.28	0.27	0.27	0.28⊕	0.21 •	0.19 •	

⊕, • statistically significant improvement or degradation vs. $\texttt{ECC}_2$

Table 7.6: $\texttt{ECC}_2$ vs. other methods: LOG-LOSS.

Dataset	$\texttt{ECC}_2$	$\texttt{EPS}_1$	$\texttt{EPS}_2$	$\texttt{RAk}_1$	$\texttt{ECC}_1$	$\texttt{EBR}_1$	$\texttt{EBR}_2$	$\texttt{RAk}_2$	IBLR	CLR
Music	3.2	3.7•	3.5 •	4.1•	3.5•	3.8•	3.5 •	4.0•	3.2	3.3
Scene	1.4	1.9•	1.6 •	2.6•	1.7•	2.1•	1.7 •	2.5•	1.3⊕	2.0 •
Yeast	11.3	12.4•	11.7 •	13.4•	12.1•	13.4•	12.7 •	14.4•	10.1⊕	10.5⊕
Genbase	0.5	0.7•	0.8 •	0.5	0.5	0.5	0.5⊕	0.5	0.7 •	
Medical	1.8	2.2•	2.1 •	2.3•	1.9•	2.0•	1.8	4.1•	3.1 •	
Slashdot	3.3	4.4•	3.8 •	5.4•	3.9•	4.4•	3.5 •	5.8•	3.7 •	
20ng	1.7	2.4•	1.9 •	3.4•	2.2•	2.5•	2.0 •	3.8•	2.8 •	3.3 •
LangLog	7.6	7.8•	7.3⊕	8.5•	8.4•	8.4•	7.4⊕	8.2•	6.8⊕	
Enron	10.2	12.3•	11.9 •	12.3•	10.9•	11.0•	10.3 •	12.9•	11.0 •	
Reuters	5.9	7.1•	6.3 •	8.1•	7.0•	7.8•	7.3 •	9.1•	5.3⊕	

⊕, • statistically significant improvement or degradation vs. $\texttt{ECC}_2$

Table 7.7: All methods: Training time (seconds).

Dataset	$\texttt{ECC}_2$	$\texttt{EPS}_1$	$\texttt{EPS}_2$	$\texttt{RAk}_1$	$\texttt{ECC}_1$	$\texttt{EBR}_1$	$\texttt{EBR}_2$	$\texttt{RAk}_2$	IBLR	CLR
Music	6	13	42	3	1	1	6	3	0	1
Scene	89	12	46	13	16	19	155	15	6	6
Yeast	150	232	461	157	32	32	163	43	4	20
Genbase	127	11	44	14	30	18	91	13	2	
Medical	544	36	87	43	51	25	92	16	2	
Slashdot	454	129	338	112	89	89	438	118	3	
20ng	6260	257	1230	1122	1553	1755	7933	2612	127	854
LangLog	1202	97	222	284	156	95	448	107	9	
Enron	593	164	226	1036	137	102	517	111	6	
Reuters	1667	780	1604	1208	185	123	628	115	53	

Table 7.8: Large datasets: Predictive Performance (with J48).

ACCURACY and per-dataset (rank) across methods

Dataset	EBR	ECC$_1$	BR	CLR	EPS$_1$	IBLR	RAk$_1$	RAk$_2$
TMC2007	0.496 (2)	**0.499** (1)	0.468 (6)	0.453 (8)	0.477 (5)	0.460 (7)	0.482 (4)	0.495 (3)
Ohsumed	0.461 (2)	**0.465** (1)	0.410 (7)	0.424 (4)	0.420 (5)	0.234 (8)	0.417 (6)	0.425 (3)
IMDB	**0.243** (1)	0.229 (3)	0.219 (4)			0.234 (2)		
Bibtex	0.332 (2)	**0.340** (1)	0.319 (3)		0.300 (4)	0.148 (7)	0.285 (6)	0.298 (5)
MediaMill	0.427 (2)	**0.429** (1)	0.362 (6)	0.415 (4)	0.426 (3)			0.412 (5)
Delicious	**0.206** (1)	0.199 (2)	0.188 (3)		0.003 (4)			

EXACT-MATCH and per-dataset (rank) across methods

Dataset	EBR	ECC$_1$	BR	CLR	EPS$_1$	IBLR	RAk$_1$	RAk$_2$
TMC2007	0.196 (3)	**0.206** (1)	0.181 (6)	0.164 (8)	0.193 (4)	0.180 (7)	0.189 (5)	0.202 (2)
Ohsumed	0.220 (2)	**0.223** (1)	0.196 (4)	0.181 (7)	0.197 (3)	0.071 (8)	0.195 (5)	0.184 (6)
IMDB	**0.054** (1)	0.049 (3)	0.045 (4)			**0.054** (1)		
Bibtex	0.108 (4)	0.125 (2)	0.116 (3)		**0.136** (1)	0.032 (7)	0.090 (6)	0.101 (5)
MediaMill	0.056 (3)	0.064 (2)	0.028 (5)	0.014 (6)	**0.071** (1)			0.053 (4)
Delicious	0.003 (3)	**0.005** (1)	0.002 (4)		0.004 (2)			

AU($\overline{\text{PRC}}$) and per-dataset (rank) across methods

Dataset	EBR	ECC$_1$	BR	CLR	EPS$_1$	IBLR	RAk$_1$	RAk$_2$
TMC2007	0.587 (2)	**0.591** (1)	0.503 (7)	0.463 (8)	0.567 (3)	0.561 (4)	0.550 (6)	0.555 (5)
Ohsumed	0.480 (2)	0.479 (3)	0.406 (7)	0.420 (6)	**0.481** (1)	0.287 (8)	0.433 (4)	0.428 (5)
IMDB	0.226 (3)	0.228 (2)	0.217 (4)			**0.253** (1)		
Bibtex	**0.337** (1)	0.334 (2)	0.290 (4)		0.293 (3)	0.145 (7)	0.286 (5)	0.262 (6)
MediaMill	**0.554** (1)	0.551 (2)	0.374 (5)	0.044 (6)	0.548 (3)			0.510 (4)
Delicious	**0.172** (1)	0.143 (2)	0.125 (3)		0.023 (4)			

F1-MACRO$^{\times L}$ and per-dataset (rank) across methods

Dataset	EBR	ECC$_1$	BR	CLR	EPS$_1$	IBLR	RAk$_1$	RAk$_2$
TMC2007	0.534 (2)	**0.535** (1)	0.517 (5)	0.455 (7)	0.489 (6)	0.436 (8)	0.518 (4)	0.530 (3)
Ohsumed	0.427 (2)	**0.430** (1)	0.398 (4)	0.408 (3)	0.376 (7)	0.111 (8)	0.380 (6)	0.390 (5)
IMDB	0.082 (2)	0.074 (3)	**0.085** (1)			0.048 (4)		
Bibtex	**0.300** (1)	0.299 (2)	0.287 (4)		0.188 (6)	0.115 (7)	0.244 (5)	0.297 (3)
MediaMill	**0.163** (1)	0.148 (4)	0.146 (5)	0.102 (6)	0.149 (3)			**0.163** (1)
Delicious	0.113 (2)	**0.127** (1)	0.085 (3)		0.004 (4)			

LOG-LOSS and per-dataset (rank) across methods

Dataset	EBR	ECC$_1$	BR	CLR	EPS$_1$	IBLR	RAk$_1$	RAk$_2$
TMC2007	5.0 (3)	4.9 (2)	5.8 (7)	5.8 (5)	5.2 (4)	**4.4** (1)	5.6 (6)	5.8 (8)
Ohsumed	5.9 (4)	5.9 (5)	5.7 (3)	5.3 (2)	5.9 (6)	**5.3** (1)	7.0 (2)	7.1 (8)
IMDB	14.2 (4)	12.3 (3)	7.5 (2)			**6.4** (1)		
Bibtex	11.0 (2)	11.1 (3)	**10.0** (1)		13.1 (6)	12.8 (5)	12.3 (4)	13.5 (7)
MediaMill	**16.1** (1)	16.2 (2)	18.9 (6)	18.7 (5)	16.7 (3)			18.3 (4)
Delicious	120.4 (2)	120.4 (3)	**109.7** (1)		152.2 (4)			

Table 7.9: Large datasets: Running time (seconds).

Dataset	EBR	ECC_1	BR	CLR	EPS_1	IBLR	RAk_1	RAk_2
TMC2007	2.6E04	2.3E04	3.4E03	6.6E03	9.4E02	4.6E02	2.0E03	1.6E04
Ohsumed	1.1E05	7.7E04	6.6E03	1.3E04	4.0E03	1.2E02	7.0E03	2.6E04
IMDB	8.3E05	8.4E05	9.3E04			4.0E03		
Bibtex	7.7E03	1.5E04	1.0E03		6.3E02	3.1E02	8.5E02	7.9E03
MediaMill	2.7E04	2.8E04	1.8E03	6.6E03	4.4E03			8.2E03
Delicious	4.1E04	6.1E05	9.1E03		1.3E02			

Chapter 8
Conclusions

This thesis addressed multi-label classification, where examples may be associated with multiple labels. In recent years sources of real-world data with multi-label associations have proliferated and this is reflected in the academic literature where the work on multi-label classification is expanding rapidly.

Multi-label data sources are progressively becoming more numerous, coming from more diverse domains, and of larger size. The focus of this research was creating general *scalable* methods; methods which:

- are generally applicable to a variety of domains;
- can compete with modern methods in the literature; and importantly
- can scale to large datasets.

This chapter briefly summarises this thesis, synthesises its contributions, and discusses future work.

8.1 Summary

Chapter 2 reviewed sources of multi-label data and introduced our dataset collection, which included new datasets; and explained how multi-label data involves an extra dimension: not only can multiple labels be associated with a single example, but many correlations exist between labels.

Chapter 3 explained the implications of the multi-label dimension on evaluation: predictive performance can either be evaluated by label or by label-set. This chapter also looked in depth into threshold functions and their calibration, and introduced a log loss measure to multi-label evaluation.

Chapter 4 reviewed and defended *problem transformation*; the paradigm for multi-label classification which was the focus of this research owing to its flexibility and general applicability, where different off-the-shelf single-label classifiers can be employed to suit requirements.

Chapter 5 reviewed significant multi-label approaches in the literature; both problem transformation methods as well as the alternative algorithm adaptation methods.

Chapter 6 presented the *pruned sets* method.

Chapter 7 presented the *classifier chains* method.

8.2 Contributions

This thesis identified a paucity in the literature on multi-label methods that are both general and scalable. Most existing methods could be identified approximately as belonging to one of two approaches. Methods of the first approach typically invest computational complexity into modelling label cor-

relations or decision boundaries and, as a result, often achieve high predictive performance, but are generally intractable on large datasets. Examples include the label combination and pairwise problem transformation methods, and numerous support vector machine and meta method adaptations. Methods of the second approach are typically efficient, and can deal with large datasets, but are specialised for a particular domain or even a particular dataset or hierarchical context, and are therefore not generally applicable. Examples include decision tree adaptations in the microbiological domain (often involved with an ontology specialisation), and probabilistic and neural-network adaptations for application to text data. Methods of this approach are either simply unable to achieve competitive predictive performance as a trade off for their emphasis on efficiency, or perform well only in a very specific context.

The major contributions of this thesis are two novel problem transformation classification methods: the *pruned sets* method (in Chapter 6) and the *classifier chains* method (in Chapter 7). These methods fill a gap in the range of available methods in the literature. They are both general and scalable methods: able to achieve high predictive performance across many domains and tractable on large datasets.

Other contributions are a new evaluation measure and datasets, and an *ensemble of binary relevance* method. The extent of our experimental evaluations also contributes generally to the depth of conclusions which can be drawn with respect to the multi-label literature.

8.2.1 Pruned Sets

The label combination method, where label sets become single-labels in the transformed problem, can be powerful on small problems since it directly models label correlations, but its ability to scale to larger problems is compromised on most datasets by the number of resulting single-labels. Furthermore, this method tends to overfit training data with relatively irregular labelling, and results in poor predictive performance.

The *pruned sets* method leverages the label skew inherent to many domains to focus on core label correlations, by pruning infrequent label sets from the data and subsampling them for frequent label sets to retain instance-space and label-correlation information. A pruned sets transformation thereby produces a single-label problem with far fewer class labels than a label combination transformation: this reduces overfitting and makes the problem much more tractable.

As compared with the label combination method, under empirical evaluation, the pruned sets method:

- reduces running times by up to two orders of magnitude; and
- achieves higher predictive performance overall.

An even greater contribution in terms of predictive power was *ensembles of pruned sets*, which:

- outperformed a variety of state-of-the-art methods from the literature in most contexts, especially under label-set-based evaluation; and
- scaled to large datasets, even when configured for maximum predictive performance.

8.2.2 Classifier Chains

The simple and intuitive binary relevance problem transformation method has often been used to scale to large datasets since it requires only a single binary model for each label. However, it had been often overlooked in the literature on account of its low predictive performance owing to its assumption of label independence.

The *classifier chains* method overcomes the label independence assumption of the binary relevance context by linking binary classifiers in a chain, along which label correlation information is passed efficiently in the attribute space of each transformed problem and, thus, unlike related methods using a meta-scheme, its complexity is bounded closely to that of the simple binary relevance method.

Thus, when compared empirically to related methods, the classifier chains method:

- has lower running time, close to that of the binary relevance method; and
- achieves superior predictive performance.

An *ensemble of classifier chains* eliminated the issue of chain order, thereby providing an even more robust and powerful method, which did not require parameter configuration other than the number of ensemble iterations. In empirical evaluation this method:

- achieved significantly better predictive performance than a variety of state-of-the-art methods, especially under label-based evaluation;

- performed well across the entire collection of datasets (unlike other methods that performed poorly on some types of data); and
- scaled to all large datasets in the collection within memory bounds.

8.2.3 Overall contributions

Pruned sets and classifier chains are both general and scalable methods. As general methods, they are more than specific algorithmic contributions, and can be employed by meta methods or be considered as meta methods themselves. As problem transformation methods they are able to embrace different learning paradigms with different base classifiers; either to adapt to a specific domain or to scale to even larger data. The ensemble frameworks presented in this thesis was example of a meta method application, to which both methods proved well suited, as the empirical evaluations showed. We report a further meta method adaptation in recent work (Read et al., 2010), which presents an implementation of the pruned sets paradigm at the leaves of an incremental decision tree learner.

The scalability and adaptability of both classifier chains and problem transformation for a variety of contexts and dimensions ensures that they can remain relevant in the evolving and expanding area of multi-label classification.

8.2.4 Other contributions

The effectiveness of ensembles in multi-label classification was further demonstrated by the contribution of *ensembles of binary relevance* models. This

method performed much better than the standalone binary relevance method, which was often used in the literature only as a baseline method, and proved suitable for very large datasets to which they scaled well and outperformed other modern methods in several contexts of experimental evaluation.

The experimental evaluations of this thesis provided, arguably, one of the most thorough empirical analyses in the multi-label literature; with respect to the number and variety of evaluation methods (including the introduction of multi-label log loss), the broad selection of classification methods, and the number and variety of datasets. This setting thus allowed the more in depth evaluation and comparison of the performance of existing methods in the literature in a variety of contexts.

The experimental framework and the methods developed during this research is freely available as open-source Java software, along with our datasets, at:

`http://meka.sourceforge.net`

8.3 Future Work

The trend to larger data sources is clear, both in the real world and the academic literature.

This thesis discussed various strategies for scaling to even larger datasets without serious degradation in predictive performance: we showed how pruned sets scale higher by more pruning, and we explained how classifier chains can eliminate redundancy in the learning space. As these methods already demonstrated a margin of predictive performance over other methods in the

literature, configurations for much greater efficiency at a small to negligible trade-off in predictive performance are acceptable for large gains in efficiency. Building hierarchical structures for these methods would further contribute to their scalability. As larger datasets become more numerous, we look forward to experimenting with such such configurations in more demanding scenarios.

Like all transformation methods, pruned sets and classifier chains can employ more efficient base classifiers for even faster running times. This thesis considered relatively demanding classification methods like support vector machines and decision trees. Future work will consider classifiers like simple and efficient perceptrons and probabilistic methods, and investigate the relationship between efficiency and relative predictive performance in these settings.

This thesis showed that training multi-label classifiers with an ever-larger number of examples begins to cause a qualitative change in predictions. Specifically, more training examples appeared to produce diminishing returns in terms of predictive performance which can be achieved by modelling label correlations explicitly. The following observations are made specifically with respect to large datasets:

- methods which model label correlations directly (e.g. pruned sets) can overfit the data more often than methods that model correlations indirectly (e.g. classifier chains);
- the extra attributes in the classifier chain transformations can begin to dominate prediction (a form of overfitting); and

- the performance of ensembles of (simple) binary relevance models can be surprisingly high.

This suggests that, given larger numbers of training examples, labels can be modelled so well individually that reducing variance by modelling label correlations is relatively less effective. Hence, as the number of examples grows ever larger, method configurations may achieve higher predictive performance by further reductions of overfitting rather than modelling label correlations. Future work will expand this issue.

In the general machine learning literature *data streams* have already gained considerable interest as an extreme context for learning, where training examples arrive continuously and rapidly and the data may exhibit concept change over time (Kirkby, 2007; Bifet et al., 2009). This type of learning environment usually entails the use of *incremental* methods, such as naive Bayes or *Hoeffding trees* (Domingos and Hulten, 2000)—an incremental tree classifier.

As scalable problem transformation methods, both pruned sets and classifier chains can be applied to the data stream scenario simply by employing incremental base classifiers. Ensemble frameworks are already commonly used for data streams (Oza and Russell, 2001) and thus the application of ensembles of pruned sets and ensembles of classifier chains in this context is natural.

It is straightforward to apply classifier chains to data stream contexts by using incremental binary base models. In an incremental context, the pruned sets method needs to be restarted at intervals to take into account variations in the label sets in the incoming examples, but this may have to be done

anyway when there is concept drift. Furthermore, the pruned sets method is particularly resistant to label-set variation, and any negative effects could be mitigated by an ensemble framework, like the one we already introduced.

We have already adapted our methods in this way to a framework for multi-label classification in data streams and carried out experiments with tens of millions of examples, and report some of the results in (Read et al., 2010). Our preliminary results are very encouraging, and we look forward to publishing further findings.

Appendix A
Nemenyi Tests

The Nemenyi tests here are additional to the corrected paired t-tests included in Chapter 6 and Chapter 7. Chapter 3 noted that this test is not as sensitive as the corrected paired t-test, however, additionally to score values, each method is given a rank, and the final rows of each table give the average rank and average value for each method. In cases where a method *did not finish* under memory bounds on a dataset (as indicated by a missing result), it should be noted that the average ranking for the method is not adversely affected by this; whereas the average value should be ignored. Note that all tables for LOG-LOSS and running-time should be read in reverse (lower numbers and higher rankings are better).

A.1 Results of the Nemenyi Test for EPS

Table A.1: EPS and state-of-the-art methods: ACCURACY.

Dataset	BR	EPS_1	RAkEL_1	EPS_2	RAkEL_2	CLR	MLkNN
Genbase	0.98 (1)	0.97 (4)	0.98 (1)	0.95 (5)	0.98 (1)		0.95 (5)
Enron	0.39 (5)	0.45 (2)	0.46 (1)	0.44 (3)	0.42 (4)		0.35 (6)
Reuters	0.32 (5)	0.50 (1)	0.45 (3)	0.50 (1)	0.30 (6)		0.44 (4)
LangLog	0.11 (6)	0.17 (2)	0.16 (3)	0.18 (1)	0.12 (4)		0.12 (4)
Music	0.50 (6)	0.57 (3)	0.58 (1)	0.57 (3)	0.58 (1)	0.54 (5)	0.40 (7)
Yeast	0.50 (7)	0.55 (1)	0.55 (1)	0.55 (1)	0.54 (4)	0.53 (6)	0.54 (4)
Medical	0.73 (5)	0.75 (2)	0.76 (1)	0.75 (2)	0.75 (2)		0.62 (6)
Slashdot	0.43 (5)	0.51 (2)	0.50 (3)	0.52 (1)	0.47 (4)		0.30 (6)
Scene	0.58 (7)	0.73 (2)	0.72 (3)	0.74 (1)	0.72 (3)	0.71 (5)	0.71 (5)
20NG	0.58 (6)	0.68 (3)	0.69 (1)	0.69 (1)	0.65 (4)	0.65 (4)	0.37 (7)
avg. rank	5.30	2.20	1.80	1.90	3.30	5.00	5.40
avg. value	0.51	0.59	0.58	0.59	0.55	0.61	0.48
Signif.	$\text{EPS}_1 \succ \text{BR}, \text{MLkNN}$; $\text{EPS}_2, \text{RAkEL}_1 \succ \text{BR}, \text{CLR}, \text{MLkNN}$;						

Table A.2: EPS and state-of-the-art methods: EXACT-MATCH.

Dataset	BR	EPS_1	RAkEL_1	EPS_2	RAkEL_2	CLR	MLkNN
Genbase	0.97 (1)	0.94 (4)	0.97 (1)	0.90 (5)	0.97 (1)		0.90 (5)
Enron	0.11 (4)	0.14 (1)	0.14 (1)	0.14 (1)	0.11 (4)		0.01 (6)
Reuters	0.27 (5)	0.38 (1)	0.36 (3)	0.38 (1)	0.25 (6)		0.29 (4)
LangLog	0.22 (4)	0.25 (2)	0.24 (3)	0.26 (1)	0.22 (4)		0.19 (6)
Music	0.26 (6)	0.32 (1)	0.31 (3)	0.32 (1)	0.31 (3)	0.27 (5)	0.13 (7)
Yeast	0.15 (7)	0.24 (1)	0.22 (3)	0.23 (2)	0.17 (5)	0.17 (5)	0.18 (4)
Medical	0.65 (2)	0.65 (2)	0.66 (1)	0.64 (5)	0.65 (2)		0.50 (6)
Slashdot	0.34 (5)	0.43 (1)	0.41 (3)	0.43 (1)	0.36 (4)		0.24 (6)
Scene	0.51 (7)	0.69 (1)	0.66 (4)	0.68 (2)	0.67 (3)	0.66 (4)	0.64 (6)
20NG	0.50 (6)	0.66 (1)	0.66 (1)	0.66 (1)	0.59 (5)	0.60 (4)	0.30 (7)
avg. rank	4.70	1.50	2.30	2.00	3.70	4.50	5.70
avg. value	0.40	0.47	0.46	0.46	0.43	0.43	0.34
Signif.	$\text{EPS}_1 \succ \text{BR}, \text{CLR}, \text{MLkNN}$; $\text{RAkEL}_1, \text{EPS}_2 \succ \text{MLkNN}$;						

Table A.3: EPS and state-of-the-art methods: $AU(\overline{PRC})$.

Dataset	BR	EPS_1	RAkEL_1	EPS_2	RAkEL_2	CLR	MLkNN
Genbase	0.98 (1)	0.96 (5)	0.98 (1)	0.95 (6)	0.98 (1)		0.97 (4)
Enron	0.36 (6)	0.45 (3)	0.46 (1)	0.43 (4)	0.42 (5)		0.46 (1)
Reuters	0.26 (5)	0.38 (3)	0.33 (4)	0.42 (2)	0.25 (6)		0.45 (1)
LangLog	0.07 (6)	0.10 (3)	0.09 (4)	0.12 (1)	0.08 (5)		0.11 (2)
Music	0.60 (6)	0.67 (3)	0.66 (4)	0.68 (1)	0.66 (4)	0.68 (1)	0.53 (7)
Yeast	0.58 (7)	0.64 (4)	0.63 (5)	0.65 (3)	0.63 (5)	0.66 (2)	0.69 (1)
Medical	0.70 (5)	0.75 (2)	0.74 (3)	0.76 (1)	0.73 (4)		0.66 (6)
Slashdot	0.37 (5)	0.50 (2)	0.42 (3)	0.54 (1)	0.40 (4)		0.27 (6)
Scene	0.62 (7)	0.78 (3)	0.72 (5)	0.81 (1)	0.72 (5)	0.76 (4)	0.81 (1)
20NG	0.56 (6)	0.73 (2)	0.65 (3)	0.77 (1)	0.62 (4)	0.62 (4)	0.44 (7)
avg. rank	5.40	3.00	3.30	2.10	4.30	2.75	3.60
avg. value	0.51	0.60	0.57	0.61	0.55	0.68	0.54
Signif.				$\text{EPS}_2 \succ$ BR;			

Table A.4: EPS and state-of-the-art methods: F1-MACRO$^{\times L}$.

Dataset	BR	EPS_1	RAkEL_1	EPS_2	RAkEL_2	CLR	MLkNN
Genbase	0.77 (1)	0.64 (4)	0.76 (2)	0.57 (6)	0.76 (2)		0.59 (5)
Enron	0.20 (3)	0.16 (4)	0.21 (1)	0.14 (5)	0.21 (1)		0.10 (6)
Reuters	0.22 (5)	0.26 (2)	0.28 (1)	0.24 (4)	0.21 (6)		0.25 (3)
LangLog	0.05 (2)	0.05 (2)	0.06 (1)	0.05 (2)	0.05 (2)		0.04 (6)
Music	0.60 (6)	0.66 (4)	0.67 (1)	0.67 (1)	0.67 (1)	0.62 (5)	0.46 (7)
Yeast	0.33 (7)	0.41 (1)	0.41 (1)	0.41 (1)	0.39 (5)	0.38 (6)	0.40 (4)
Medical	0.35 (2)	0.31 (4)	0.36 (1)	0.29 (5)	0.35 (2)		0.23 (6)
Slashdot	0.34 (3)	0.33 (4)	0.35 (1)	0.33 (4)	0.35 (1)		0.16 (6)
Scene	0.68 (7)	0.76 (1)	0.75 (3)	0.76 (1)	0.75 (3)	0.74 (6)	0.75 (3)
20NG	0.66 (6)	0.71 (2)	0.70 (3)	0.72 (1)	0.69 (5)	0.70 (3)	0.43 (7)
avg. rank	4.20	2.80	1.50	3.00	2.80	5.00	5.30
avg. value	0.42	0.43	0.45	0.42	0.44	0.61	0.34
Signif.				$\text{RAkEL}_1 \succ$ CLR; MLkNN;			

Table A.5: EPS and state-of-the-art methods: LOG-LOSS.

Dataset	BR	EPS_1	RAkEL_1	EPS_2	RAkEL_2	CLR	MLkNN
Genbase	0.5 (5)	0.7 (3)	0.5 (4)	0.8 (1)	0.5 (5)		0.8 (2)
Enron	14.5 (1)	12.3 (4)	12.4 (3)	11.9 (5)	12.9 (2)		10.3 (6)
Reuters	9.0 (2)	7.1 (4)	8.2 (3)	6.3 (5)	9.1 (1)		4.6 (6)
LangLog	8.1 (3)	7.8 (4)	8.6 (1)	7.3 (5)	8.2 (2)		5.2 (6)
Music	5.9 (1)	3.7 (5)	4.1 (2)	3.5 (6)	4.0 (3)	3.3 (7)	3.7 (4)
Yeast	16.3 (1)	12.4 (4)	13.3 (3)	11.7 (5)	14.0 (2)	10.5 (6)	10.2 (7)
Medical	2.6 (1)	2.2 (5)	2.3 (4)	2.1 (6)	2.4 (3)		2.5 (2)
Slashdot	6.4 (1)	4.4 (4)	5.6 (3)	3.8 (5)	5.8 (2)		3.7 (6)
Scene	3.8 (1)	1.9 (5)	2.6 (2)	1.6 (6)	2.5 (3)	2.0 (4)	1.4 (7)
20NG	4.6 (1)	2.4 (6)	3.4 (3)	1.9 (7)	3.7 (2)	3.3 (4)	3.0 (5)
avg. rank	1.70	4.40	2.80	5.10	2.50	5.25	5.10
avg. value	7.18	5.49	6.10	5.08	6.32	4.78	4.54
Signif.	$\text{BR} \succ \text{EPS}_2, \text{CLR}, \text{MLkNN}$;						

Table A.6: EPS and state-of-the-art methods: RUNNING-TIME (seconds).

Dataset	BR	EPS_1	RAkEL_1	EPS_2	RAkEL_2	CLR	MLkNN
Genbase	2 (5)	11 (4)	14 (2)	44 (1)	11 (3)		2 (6)
Enron	21 (5)	164 (3)	1004 (1)	226 (2)	163 (4)		2 (6)
Reuters	22 (5)	780 (3)	1210 (2)	1604 (1)	190 (4)		4 (6)
LangLog	17 (5)	97 (4)	255 (1)	222 (2)	124 (3)		3 (6)
Music	0 (6)	13 (2)	3 (3)	42 (1)	2 (4)	0 (5)	0 (7)
Yeast	6 (6)	232 (2)	99 (3)	461 (1)	24 (4)	12 (5)	1 (7)
Medical	3 (5)	36 (2)	32 (3)	87 (1)	13 (4)		0 (6)
Slashdot	17 (5)	129 (2)	83 (4)	338 (1)	95 (3)		1 (6)
Scene	6 (5)	12 (4)	14 (3)	46 (1)	14 (2)	5 (6)	3 (7)
20NG	507 (5)	257 (6)	1097 (3)	1230 (2)	3371 (1)	854 (4)	98 (7)
avg. rank	5.20	3.20	2.50	1.30	3.20	5.00	6.40
avg. value	60	173	381	430	401	218	11
Signif.	$\text{EPS}_1, \text{RAkEL}_1, \text{RAkEL}_2 \succ \text{MLkNN}$; $\text{EPS}_2 \succ \text{BR}, \text{CLR}, \text{MLkNN}$						

A.2 Results of the Nemenyi Test for ECC

Table A.7: ECC and state-of-the-art methods: ACCURACY.

Dataset	$\texttt{ECC}_2$	$\texttt{RAkEL}_1$	$\texttt{ECC}_1$	$\texttt{EBR}_1$	$\texttt{EBR}_2$	$\texttt{RAkEL}_2$	IBLR	CLR
Genbase	0.98 (1)	0.98 (1)	0.98 (1)	0.98 (1)	0.98 (1)	0.98 (1)	0.95 (7)	
LangLog	0.17 (2)	0.16 (3)	0.14 (4)	0.14 (4)	0.18 (1)	0.11 (6)	0.04 (7)	
Music	0.56 (3)	0.57 (2)	0.55 (5)	0.55 (5)	0.55 (5)	0.58 (1)	0.56 (3)	0.54 (8)
Medical	0.79 (1)	0.76 (5)	0.78 (2)	0.77 (4)	0.78 (2)	0.52 (7)	0.54 (6)	
Slashdot	0.52 (2)	0.51 (3)	0.51 (3)	0.51 (3)	0.53 (1)	0.47 (6)	0.26 (7)	
Yeast	0.54 (1)	0.54 (1)	0.53 (4)	0.53 (4)	0.53 (4)	0.53 (4)	0.54 (1)	0.53 (4)
Enron	0.47 (1)	0.46 (3)	0.46 (3)	0.46 (3)	0.47 (1)	0.42 (6)	0.34 (7)	
20NG	0.69 (2)	0.69 (2)	0.68 (4)	0.68 (4)	0.70 (1)	0.65 (6)	0.39 (8)	0.65 (6)
Reuters	0.49 (1)	0.45 (3)	0.46 (2)	0.37 (6)	0.41 (4)	0.30 (7)	0.38 (5)	
Scene	0.71 (4)	0.72 (2)	0.70 (7)	0.69 (8)	0.71 (4)	0.72 (2)	0.73 (1)	0.71 (4)
avg. rank	1.80	2.50	3.50	4.20	2.40	4.60	5.20	5.50
avg. value	0.59	0.58	0.58	0.57	0.58	0.53	0.47	0.61
Signif.	$\texttt{ECC}_2 \succ \texttt{IBLR}$; $\texttt{ECC}_2 \succ \texttt{CLR}$;							

Table A.8: ECC and state-of-the-art methods: $AU(\overline{PRC})$.

Dataset	$\texttt{ECC}_2$	$\texttt{RAkEL}_1$	$\texttt{ECC}_1$	$\texttt{EBR}_1$	$\texttt{EBR}_2$	$\texttt{RAkEL}_2$	IBLR	CLR
Genbase	0.99 (1)	0.98 (2)	0.98 (2)	0.98 (2)	0.98 (2)	0.98 (2)	0.96 (7)	
LangLog	0.10 (2)	0.09 (3)	0.08 (4)	0.08 (4)	0.11 (1)	0.07 (6)	0.06 (7)	
Music	0.69 (1)	0.66 (8)	0.68 (3)	0.67 (6)	0.68 (3)	0.67 (6)	0.69 (1)	0.68 (3)
Medical	0.81 (1)	0.74 (5)	0.79 (3)	0.78 (4)	0.80 (2)	0.51 (7)	0.59 (6)	
Slashdot	0.57 (1)	0.43 (5)	0.52 (3)	0.48 (4)	0.54 (2)	0.40 (6)	0.29 (7)	
Yeast	0.67 (2)	0.63 (7)	0.66 (3)	0.64 (5)	0.64 (5)	0.62 (8)	0.69 (1)	0.66 (3)
Enron	0.53 (1)	0.46 (5)	0.51 (3)	0.51 (3)	0.52 (2)	0.42 (7)	0.45 (6)	
20NG	0.79 (1)	0.65 (2)				0.62 (3)		0.62 (3)
Reuters	0.42 (1)	0.33 (6)	0.37 (3)	0.34 (5)	0.37 (3)	0.25 (7)	0.40 (2)	
Scene	0.81 (2)	0.72 (7)	0.78 (3)	0.73 (6)	0.77 (4)	0.72 (7)	0.82 (1)	0.76 (5)
avg. rank	1.30	5.00	3.00	4.33	2.67	5.90	4.22	3.50
avg. value	0.64	0.57	0.60	0.58	0.60	0.53	0.55	0.68
Signif.	$\texttt{ECC}_2 \succ \texttt{RAkEL}_1$; $\texttt{ECC}_2 \succ \texttt{RAkEL}_2$;							

Table A.9: ECC and state-of-the-art methods: EXACT-MATCH.

Dataset	$\texttt{ECC}_2$	$\texttt{RAkEL}_1$	$\texttt{ECC}_1$	$\texttt{EBR}_1$	$\texttt{EBR}_2$	$\texttt{RAkEL}_2$	IBLR	CLR
Genbase	0.95 (3)	0.96 (2)	0.95 (3)	0.95 (3)	0.95 (3)	0.97 (1)	0.91 (7)	
LangLog	0.25 (1)	0.24 (3)	0.22 (4)	0.22 (4)	0.25 (1)	0.21 (6)	0.14 (7)	
Music	0.29 (4)	0.31 (2)	0.27 (5)	0.26 (8)	0.27 (5)	0.32 (1)	0.30 (3)	0.27 (5)
Medical	0.69 (1)	0.67 (4)	0.68 (2)	0.67 (4)	0.68 (2)	0.45 (6)	0.42 (7)	
Slashdot	0.42 (1)	0.42 (1)	0.42 (1)	0.42 (1)	0.42 (1)	0.36 (6)	0.15 (7)	
Yeast	0.19 (2)	0.22 (1)	0.19 (2)	0.15 (7)	0.15 (7)	0.16 (6)	0.19 (2)	0.17 (5)
Enron	0.12 (2)	0.14 (1)	0.12 (2)	0.11 (5)	0.12 (2)	0.11 (5)	0.02 (7)	
20NG	0.64 (3)	0.66 (1)	0.63 (4)	0.63 (4)	0.65 (2)	0.59 (7)	0.32 (8)	0.60 (6)
Reuters	0.34 (2)	0.36 (1)	0.32 (3)	0.26 (5)	0.29 (4)	0.25 (6)	0.25 (6)	
Scene	0.65 (5)	0.66 (2)	0.64 (7)	0.63 (8)	0.65 (5)	0.67 (1)	0.66 (2)	0.66 (2)
avg. rank	2.40	1.80	3.30	4.90	3.20	4.50	5.60	4.50
avg. value	0.45	0.46	0.44	0.43	0.44	0.41	0.34	0.43
Signif.				$\texttt{RAkEL}_1 \succ \texttt{IBLR}$;				

Table A.10: ECC and state-of-the-art methods: F1-MACRO$^{\times L}$.

Dataset	$\texttt{EBR}_2$	$\texttt{RAkEL}_1$	$\texttt{ECC}_1$	$\texttt{EBR}_1$	$\texttt{RAkEL}_2$	$\texttt{ECC}_2$	IBLR	CLR
Genbase	0.75 (3)	0.76 (1)	0.75 (3)	0.75 (3)	0.76 (1)	0.75 (3)	0.63 (7)	
LangLog	0.06 (1)	0.06 (1)	0.05 (4)	0.05 (4)	0.05 (4)	0.06 (1)	0.02 (7)	
Music	0.65 (5)	0.67 (1)	0.65 (5)	0.64 (7)	0.67 (1)	0.66 (3)	0.66 (3)	0.62 (8)
Medical	0.35 (1)	0.35 (1)	0.35 (1)	0.35 (1)	0.25 (6)	0.35 (1)	0.20 (7)	
Slashdot	0.36 (1)	0.35 (3)	0.34 (5)	0.34 (5)	0.35 (3)	0.36 (1)	0.15 (7)	
Yeast	0.37 (5)	0.41 (1)	0.37 (5)	0.37 (5)	0.38 (3)	0.37 (5)	0.41 (1)	0.38 (3)
Enron	0.20 (3)	0.21 (1)	0.20 (3)	0.20 (3)	0.21 (1)	0.20 (3)	0.13 (7)	
20NG	0.73 (1)	0.70 (5)	0.71 (3)	0.71 (3)	0.69 (7)	0.73 (1)	0.45 (8)	0.70 (5)
Reuters	0.28 (1)	0.28 (1)	0.27 (3)	0.27 (3)	0.21 (6)	0.27 (3)	0.19 (7)	
Scene	0.74 (4)	0.74 (4)	0.73 (7)	0.73 (7)	0.75 (1)	0.75 (1)	0.75 (1)	0.74 (4)
avg. rank	2.50	1.90	3.90	4.10	3.30	2.20	5.50	5.00
avg. value	0.45	0.45	0.44	0.44	0.43	0.45	0.36	0.61
Signif.				$\texttt{RAkEL}_1 \succ \texttt{IBLR}$;				

Table A.11: ECC and state-of-the-art methods: Log-loss.

Dataset	$\texttt{ECC}_2$	$\texttt{RAkEL}_1$	$\texttt{ECC}_1$	$\texttt{EBR}_1$	$\texttt{EBR}_2$	$\texttt{RAkEL}_2$	IBLR	CLR
Genbase	0.5 (4)	0.5 (2)	0.5 (2)	0.5 (4)	0.5 (4)	0.5 (4)	0.7 (1)	
LangLog	7.6 (5)	8.5 (1)	8.4 (2)	8.4 (3)	7.4 (6)	8.2 (4)	6.8 (7)	
Music	3.2 (7)	4.1 (1)	3.5 (5)	3.8 (3)	3.5 (4)	4.0 (2)	3.2 (8)	3.3 (6)
Medical	1.8 (6)	2.3 (3)	1.9 (5)	2.0 (4)	1.8 (6)	4.1 (1)	3.1 (2)	
Slashdot	3.3 (7)	5.4 (2)	3.9 (4)	4.4 (3)	3.5 (6)	5.8 (1)	3.7 (5)	
Yeast	11.3 (6)	13.4 (3)	12.1 (5)	13.4 (2)	12.7 (4)	14.4 (1)	10.1 (8)	10.5 (7)
Enron	10.2 (7)	12.3 (2)	10.9 (5)	11.0 (3)	10.3 (6)	12.9 (1)	11.0 (3)	
20NG	1.7 (8)	3.4 (2)	2.2 (6)	2.5 (5)	2.0 (7)	3.8 (1)	2.8 (4)	3.3 (3)
Reuters	5.9 (6)	8.1 (2)	7.0 (5)	7.8 (3)	7.3 (4)	9.1 (1)	5.3 (7)	
Scene	1.4 (7)	2.6 (1)	1.7 (6)	2.1 (3)	1.7 (5)	2.5 (2)	1.3 (8)	2.0 (4)
avg. rank	6.30	1.90	4.50	3.30	5.20	1.80	5.30	5.00
avg. value	4.69	6.08	5.21	5.60	5.07	6.52	4.80	4.78
Signif.	$\texttt{RAkEL}_1 \succ \texttt{ECC}_2$; $\texttt{RAkEL}_1 \succ$ IBLR; RAkEL;							

Table A.12: ECC and state-of-the-art methods: Running-time (seconds).

Dataset	$\texttt{ECC}_2$	$\texttt{RAkEL}_1$	$\texttt{ECC}_1$	$\texttt{EBR}_1$	$\texttt{EBR}_2$	$\texttt{RAkEL}_2$	IBLR	CLR
Genbase	127 (1)	14 (5)	30 (3)	18 (4)	91 (2)	13 (6)	2 (7)	
LangLog	1202 (1)	284 (3)	156 (4)	95 (6)	448 (2)	107 (5)	9 (7)	
Music	6 (1)	3 (4)	1 (5)	1 (6)	6 (2)	3 (3)	0 (8)	1 (7)
Medical	544 (1)	43 (4)	51 (3)	25 (5)	92 (2)	16 (6)	2 (7)	
Slashdot	454 (1)	112 (4)	89 (6)	89 (5)	438 (2)	118 (3)	3 (7)	
Yeast	150 (3)	157 (2)	32 (5)	32 (6)	163 (1)	43 (4)	4 (8)	20 (7)
Enron	593 (2)	1036 (1)	137 (4)	102 (6)	517 (3)	111 (5)	6 (7)	
20NG	6260 (2)	1122 (6)	1553 (5)	1755 (4)	7933 (1)	2612 (3)	127 (9)	854 (7)
Reuters	1667 (1)	1208 (2)	185 (4)	123 (5)	628 (3)	115 (6)	53 (7)	
Scene	89 (2)	13 (6)	16 (4)	19 (3)	155 (1)	15 (5)	6 (8)	6 (7)
avg. rank	1.50	3.70	4.30	5.00	1.90	4.60	7.50	7.00
avg. value	1109	399	225	226	1047	315	21	220
Signif.	$\texttt{EBR}_2$; $\texttt{ECC}_2$; $\texttt{RAkEL}_1 \succ$ IBLR; CLR; BR							

Bibliography

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., and Sherlock, G. (2000). Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29.

Bade, K., Hüllermeier, E., and Nürnberger, A. (2006). Hierarchical classification by expected utility maximization. In *ICDM '06: Sixth International Conference on Data Mining*, pages 43–52, Washington, DC, USA. IEEE Computer Society.

Barutcuoglu, Z., Schapire, R. E., and Troyanskaya, O. G. (2006). Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., and Gavaldà, R. (2009). New ensemble methods for evolving data streams. In *KDD '09: 15th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. ACM.

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

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Cai, L. (2008). *Multilabel Classification over Category Taxonomies*. PhD thesis, Department of Computer Science, Brown University, Providence, Rhode Island.

Cesa-Bianchi, N., Gentile, C., and Zaniboni, L. (2006). Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research*, 7:31–54.

Chan, A. and Freitas, A. A. (2006). A new ant colony algorithm for multi-label classification with applications in bioinformatics. In *GECCO '06: 8th Annual Conference on Genetic and Evolutionary Computation*, pages 27–34, New York, NY, USA. ACM Press.

Cheng, W. and Hüllermeier, E. (2009). Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225.

Chi, E. H. and Mytkowicz, T. (2007). Understanding navigability of social tagging systems. In *Computer/Human Interaction*.

Clare, A. and King, R. D. (2001). Knowledge discovery in multi-label phenotype data. *Lecture Notes in Computer Science*, 2168.

Cohen, W. W. (1995). Fast effective rule induction. In *ICML'95: Twelfth*

International Conference on Machine Learning, pages 115–123. Morgan Kaufmann.

Crammer, K. and Singer, Y. (2003). A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, 3:1025–1058.

De Comité, F., Gilleron, R., and Tommasi, M. (2003). Learning multi-label alternating decision trees from texts and data. *Machine Learning and Data Mining in Pattern Recognition*, pages 251–274.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

Dimou, A., Tsoumakas, G., Mezaris, V., Kompatsiaris, I., and Vlahavas, I. (2009). An empirical study of multi-label learning methods for video annotation. In *Proceedings of the 7th International Workshop on Content-Based Multimedia Indexing*. IEEE.

Diplaris, S., Tsoumakas, G., Mitkas, P., and Vlahavas, I. (2005). Protein classification with multiple algorithms. In *PCI '05: 10th Panhellenic Conference on Informatics*, pages 448–456. MIT Press.

Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In *KDD '00: 6th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 71–80.

Elisseeff, A. and Weston, J. (2001). A kernel method for multi-labelled classification. In *In Advances in Neural Information Processing Systems 14*, pages 681–687. MIT Press.

Fan, R.-E. and Lin, C. J. (2007). A study on threshold selection for multi-label classification. Technical report, National Taiwan University.

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

Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., and Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine Learning*, 73(2):133–153.

Gelbukh, A. and Sidorov, G. (2001). Zipf and heaps laws' coefficients depend on language. In *CICLing '01: Second International Conference on Computational Linguistics and Intelligent Text Processing*, page 332, Berlin, Heidelberg. Springer.

Ghamrawi, N. and McCallum, A. (2005). Collective multi-label classification. In *CIKM '05: 14th ACM international Conference on Information and Knowledge Management*, pages 195–200, New York, NY, USA. ACM Press.

Godbole, S. and Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. In *PAKDD '04: Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 22–30. Springer.

Godbole, S., Sarawagi, S., and Chakrabarti, S. (2002). Scaling multi-class support vector machines using inter-class confusion. In *KDD '02: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 513–518.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Peter, R., and Witten, I. H.

(2009). The weka data mining software: An update. *SIGKDD Explorations*, 11(1).

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Hersh, W., Buckley, C., Leone, T. J., and Hickam, D. (1994). Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *SIGIR '94: 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 192–201, New York, NY, USA. Springer.

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

Ji, S., Tang, L., Yu, S., and Ye, J. (2008). Extracting shared subspace for multi-label classification. In *KDD '08: 14th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 381–389. ACM.

Jiang, A., Wang, C., and Zhu, Y. (2008). Calibrated Rank-SVM for multi-label image categorization. In *IJCNN '08: IEEE International Joint Conference on Neural Networks, 2008*, pages 1450–1455.

John, G. H. and Langley, P. (1995). Estimating continuous distributions in Bayesian classifiers. In *UAI '95 : 11th Conference on Uncertainty in Artificial Intelligence*, pages 338–345.

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

Kirkby, R. (2007). *Improving Hoeffding Trees*. PhD thesis, Department of Computer Science, University of Waikato.

Labrou, Y. and Finin, T. (1999). Yahoo! as an ontology: using yahoo! categories to describe documents. In *CIKM '99: Eighth International Conference on Information and Knowledge Management*, pages 180–187, New York, NY, USA. ACM Press.

Lang, K. (2008). The 20 newsgroups dataset. "`http://people.csail.mit.edu/jrennie/20Newsgroups/`".

Lewis, D. D. (1992). An evaluation of phrasal and clustered representations on a text categorization task. In *SIGIR '92: 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–50, New York, NY, USA. ACM.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In *ECML '98: 10th European Conference on Machine Learning*, pages 4–15. Springer.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *Machine Learning*, 5:361–397.

Loza Mencía, E. and Fürnkranz, J. (2008). Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *ECML-PKDD*

'08: European Conference on Machine Learning and Knowledge Discovery in Databases, pages 50–65. Springer.

Luo, X. and Zincir-Heywood, N. A. (2005). Evaluation of two systems on multi-class multi-label document classification. In *ISMIS '05: 16th International Symposium on Methodologies for Intelligent Systems*, pages 161–169.

Maron, O., Lozano-Prez, T., and p Erez, T. A. L. (1998). A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems*, pages 570–576. MIT Press.

McCallum, A., Rosenfeld, R., Mitchell, T., and Ng, A. (1998). Improving text classification by shrinkage in a hierarchy of classes. In *ICML '98: 15th International Conference on Machine Learning*, pages 359–367.

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

Medelyan, O. (2009). *Human-competitive automatic topic indexing*. PhD thesis, The University of Waikato, Waikato, New Zealand.

Mencía, E. L. and Fürnkranz, J. (2008). Pairwise learning of multilabel classifications with perceptrons. In *IJCNN '08: International Joint Conference on Neural Networks*, pages 2899–2906.

Mewes, H. W., Albermann, K., Heumann, K., Liebl, S., and Pfeiffer, F. (1997). MIPS: a database for protein sequences, homology data and yeast genome information. *Nucleic Acids Research*, 25(1):28–30.

Mladenic, D. and Grobelnik, M. (1998). Feature selection for classification based on text hierarchy. In *CONALD '98: Conference on Automatic Learning and Discovery*.

Nadeau, C. and Bengio, Y. (2003). Inference for the generalization error. *Machine Learning*, 52(3):239–281.

Oliveira, E., Ciarelli, P. M., Badue, C., and De Souza, A. F. (2008). A comparison between a KNN based approach and a PNN algorithm for a multi-label classification problem. In *ISDA '08: Eighth International Conference on Intelligent Systems Design and Applications*, volume 2, pages 628–633, Washington, DC, USA. IEEE Computer Society.

Oza, N. and Russell, S. (2001). Online bagging and boosting. In *Artificial Intelligence and Statistics 2001*, pages 105–112. Morgan Kaufmann.

Park, S.-H. and Fürnkranz, J. (2008). Multi-label classification with label constraints. Technical report, Knowledge Engineering Group, TU Darmstadt.

Pestian, J., Brew, C., Matykiewicz, P., Hovermale, D., Johnson, N., Cohen, K. B., and Duch, W. (2007). A shared task involving multi-label classification of clinical free text. In ACL, editor, *Proceedings of ACL BioNLP*, Prague. Association of Computational Linguistics.

Petrovskiy, M. (2006). Paired comparisons method for solving multi-label learning problem. In *HIS '06: Sixth International Conference on Hybrid Intelligent Systems*. IEEE.

Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Ráez, A. M., López, L. A. U., and Steinberger, R. (2004). Adaptive selection of base classifiers in one-against-all learning for large multi-labeled collections. In *EsTAL: 4th International Conference on Advances in Natural Language Processing*, pages 1–12.

Read, J. (2008). A pruned problem transformation method for multi-label classification. In *NZCSRS 08: 2008 New Zealand Computer Science Research Student Conference*, pages 143–150.

Read, J., Bifet, A., Holmes, G., and Pfahringer, B. (2010). Efficient multi-label classification for evolving data streams. Technical report, University of Waikato, Hamilton, New Zealand. Working Paper 2010/04.

Read, J., Pfahringer, B., and Holmes, G. (2008). Multi-label classification using ensembles of pruned sets. In *ICDM'08: Eighth IEEE International Conference on Data Mining*, pages 995–1000. IEEE.

Read, J., Pfahringer, B., and Holmes, G. (2009a). Generating synthetic multi-label data streams. In *MLD '09: 1st ECML/PKDD 2009 Workshop on Learning from Multi-Label Data*.

Read, J., Pfahringer, B., Holmes, G., and Frank, E. (2009b). Classifier chains

for multi-label classification. In *ECML '09: 20th European Conference on Machine Learning*, pages 254–269. Springer.

Ruepp, A., Zollner, A., Maier, D., Albermann, K., Hani, J., Mokrejs, M., Tetko, I., Güldener, U., Mannhaupt, G., Münsterkötter, M., and Mewes, H. W. (2004). The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Research*, 32(18):5539–5545.

Sapozhnikova, E. P. (2009). Multi-label classification with art neural networks. In *WKDD '09: Second International Workshop on Knowledge Discovery and Data Mining, 2009.*, pages 144–147.

Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

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

Snoek, C. G. M., Worring, M., van Gemert, J. C., Geusebroek, J.-M., and Smeulders, A. W. M. (2006). The challenge problem for automated detection of 101 semantic concepts in multimedia. In *MULTIMEDIA '06: 14th annual ACM international conference on Multimedia*, pages 421–430, New York, NY, USA. ACM.

Spyromitros, E., Tsoumakas, G., and Vlahavas, I. (2008). An empirical study of lazy multilabel classification algorithms. In *SETN '08: Fifth Hellenic conference on Artificial Intelligence*, pages 401–406, Berlin, Heidelberg. Springer.

Srivastava, A. and Zane-Ulman, B. (2005). Discovering recurring anomalies in text reports regarding complex space systems. In *IEEE Aerospace Conference 2005*.

Streich, A. and Buhmann, J. (2008). Classification of multi-labeled data: A generative approach. In *ECML-PKDD '08: 19th European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 390–405.

Struyf, J., Dzeroski, S., Blockeel, H., and Clare, A. (2005). 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*, pages 272–283. Springer.

Sun, L., Ji, S., and Ye, J. (2008). Hypergraph spectral learning for multi-label classification. In *KDD '08: 14th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 668–676. ACM.

Tan, P. J. and Dowe, D. L. (2003). MML inference of decision graphs with multi-way joins and dynamic attributes. In *AI '09: 22nd Australian Conference on Artificial Intelligence*, pages 269–281.

Tang, L., Rajan, S., and Narayanan, V. K. (2009). 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. ACM.

Thabtah, F. A., Cowling, P., and Peng, Y. (2004). MMAC: A new multi-class, multi-label associative classification approach. In *ICDM '04: Fourth IEEE International Conference on Data Mining*, pages 217–224.

Trohidis, K., Tsoumakas, G., Kalliris, G., and Vlahavas, I. (2008). Multilabel classification of music into emotions. In *ISMIR '08: 9th International Conference on Music Information Retrieval*.

Tsoumakas, G., Dimou, A., Spyromitros, E., Mezaris, V., Kompatsiaris, I., and Vlahavas, I. (2009a). Correlation-based pruning of stacked binary relevance models for multi-label learning. In *MLD '09: 1st ECML/PKDD Workshop on Learning from Multi-Label Data*, pages 101–116.

Tsoumakas, G. and Katakis, I. (2007). Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13.

Tsoumakas, G., Katakis, I., and Vlahavas, I. (2010). Mining multi-label data. In Maimon, O. and Rokach, L., editors, *Data Mining and Knowledge Discovery Handbook*. 2nd edition, Springer.

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

Tsoumakas, G., Vilcek, J., Xioufis, E. S., and et al. (2009b). MULAN: A java library for multi-label learning. "`http://mlkd.csd.auth.gr/multilabel.html#Software`".

Tsoumakas, G. and Vlahavas, I. P. (2007). Random k-labelsets: An ensemble method for multilabel classification. In *ECML '07: 18th European Conference on Machine Learning*, pages 406–417. Springer.

Ueda, N. and Saito, K. (2002). Parametric mixture models for multi-labeled text. In *NIPS '02: Neural Information Processing Systems 2002*, pages 721–728.

Veloso, A., Meira, Jr., W., Gonçalves, M., and Zaki, M. (2007). Multi-label lazy associative classification. In *PKDD '07: 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, pages 605–612, Berlin, Heidelberg. Springer.

Vens, C., Struyf, J., Schietgat, L., Džeroski, S., and Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Machine Learning*, 2(73):185–214.

Vilar, D. and José, M. (2004). Multi-label text classification using multinomial models. In *EsTAL '04: Fourth International Conference of España for Natural Language Processing*.

Wang, J., Zucker, and Jean-Daniel (2000). Solving multiple-instance problem: A lazy learning approach. In *ICML '00: 17th International Conference on Machine Learning*, pages 1119–1125.

Wang, K., Zhou, S., and Liew, S. C. (1999). Building hierarchical classifiers using class proximity. In Atkinson, M. P., Orlowska, M. E., Valduriez, P., Zdonik, S. B., and Brodie, M. L., editors, *VLDB '99: 25th International Conference on Very Large Data Bases*, pages 363–374, Edinburgh, UK. Morgan Kaufmann Publishers, San Francisco, US.

Wibowo, W. and Williams, H. E. (2002). Simple and accurate feature selection for hierarchical categorisation. In *DocEng '02: 2002 ACM Sym-*

posium on Document Engineering, pages 111–118, New York, NY, USA. ACM Press.

Yan, R., Tesic, J., and Smith, J. R. (2007). Model-shared subspace boosting for multi-label classification. In *KDD '07: 13th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 834–843. ACM.

Yang, Y. (2001). A study on thresholding strategies for text categorization. In *Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval*, pages 137–145. ACM Press.

Zha, Z.-J., Hua, X.-S., Mei, T., Wang, J., Qi, G.-J., and Wang, Z. (2008). Joint multi-label multi-instance learning for image classification. In *CVPR '08: IEEE Conference on Computer Vision and Pattern Recognition, 2008.*

Zhang, M.-L., Peña, J. M., and Robles, V. (2009). Feature selection for multi-label naive Bayes classification. *Inf. Sci.*, 179(19):3218–3229.

Zhang, M.-L. and Zhou, Z.-H. (2005). A k-nearest neighbor based algorithm for multi-label classification. In *GnC '05: IEEE International Conference on Granular Computing, 2005*, pages 718–721. IEEE.

Zhang, M.-L. and Zhou, Z.-H. (2006). Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351.

Zhang, M.-L. and Zhou, Z.-H. (2007a). ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048.

Zhang, M. L. and Zhou, Z. H. (2007b). Multi-label learning by instance differentiation. In *(AAAI'08): 23rd AAAI Conference on Artificial Intelligence*, pages 669–674. AAAI Press.

Zhou, Z. H. and Zhang, M. L. (2006). Multi-instance multi-label learning with application to scene classification. In *NIPS '06: Neural Information Processing Systems 2006*, pages 1609–1616. MIT Press.

Zhu, S., Ji, X., Xu, W., and Gong, Y. (2005). Multi-labelled classification using maximum entropy method. In *SIGIR: '05: 27th Annual ACM Conference on Research and Development in Information Retrieval*, pages 274–281.