Examining Label Intersections in Pairwise Multilabel Classification

Untersuchung von Label-Schnittmengen in paarweiser Multilabel-Klassifizierung

Bachelor-Thesis von Tomasz Gasiorowski
September 2015

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Knowledge Engineering Group

Examining Label Intersections in Pairwise Multilabel Classification
Untersuchung von Label-Schnittmengen in paarweiser Multilabel-Klassifizierung

vorgelegte Bachelor-Thesis von Tomasz Gasiorowski

1. Gutachten: Prof. Dr. Johannes Fürnkranz

2. Gutachten: Dr. Eneldo Loza Mencia

Tag der Einreichung: 02.09.2015

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 02. September 2015

(Tomasz Gasiorowski)

Abstract

Due to the overwhelming amount of data being processed nowadays, the importance of automated data classification is rising. Classifiers are relevant for many industries as they handle topics such as medical image analysis, internet search queries and email spam filtering. Typical tasks can be solved through multiclass classification, which involves assigning one of multiple classes to an instance. A variant of the traditional multiclass classification problem is multilabel classification. In this setting, classes are not mutually exclusive and samples can belong to multiple classes simultaneously. The dependencies between classes, particularly their overlapping areas, is the reason why this is a challenging problem. Classification through pairwise decomposition is one of the leading methods for solving multilabel classification. In this method we transform the multilabel problem into single-label problems through learning classifiers for each pair of labels and combining their outputs to receive the end result. In this thesis we will implement a modified pairwise decomposition method for multilabel classification and compare its results to those of other approaches. We will also go deeper into the analysis of overlapping areas and how this information can be used to make optimizations.

Contents

1 Introduction **11**

1.1 Motivation . . . 11
1.2 Multiclass Classification . . . 11
1.3 Multilabel Case . . . 11
1.4 Outline . . . 12

2 Multilabel Classification **13**

2.1 Definitions . . . 13
2.2 Baseline Approaches . . . 14
2.2.1 Binary Relevance . . . 14
2.2.2 Pairwise Decomposition . . . 14
2.2.3 Support Vector Machines (SVM) . . . 15
2.2.4 Label Powerset . . . 16
2.2.5 Decision trees . . . 16
2.2.6 Voting systems . . . 17

3 Related Work **19**

4 Pairwise with Intersections (PWI) **21**

4.1 Training . . . 21
4.2 Voting . . . 21
4.2.1 Binary voting . . . 22
4.2.2 Weighted and hybrid voting . . . 23

5 Triple Class Pairwise (TCP) **25**

5.1 Training . . . 25
5.2 Voting . . . 25
5.3 Outlook . . . 26

6 Confusion Matrix Analysis **27**

6.1 Parent-child problem . . . 29
6.2 Class pairs without intersections . . . 29

7 Experiments **31**

7.1 Measures . . . 31
7.2 Base Classifiers J48 . . . 32

7.3 Datasets 33
7.4 Evaluation 33
7.4.1 Pairwise with Intersections 34
7.4.2 Parent-child 37

8 Conclusion and Future Work **43**

List of Figures

1.1 Overlapping classes (Wikipedia, 2015) 12

2.1 Support Vector Machine (Law, 2011) 16
2.2 Decision tree 17

4.1 PWI training 21
4.2 PWI voting cases 22

7.1 J48 model 33
7.2 J48 model label 2 vs label 1 \ label 2 emotions 38
7.3 J48 model label 1 vs label 2 \ label 1 emotions 38
7.4 J48 model label 1 vs label 2 \ label 1 yeast 40
7.5 J48 models for (label 11, label 12) yeast 40
7.6 J48 models for (label 9, label 11) genbase 42
7.7 J46 models for (label 6, label 5) genbase 42

List of Tables

4.1 PWI binary voting 23
4.2 PWI weighted voting 24
4.3 PWI hybrid voting 24

5.1 TCP binary voting 25
5.2 TCP weighted voting 26

6.1 Confusion matrix structure 27
6.2 Confusion matrix example 28

7.1 Datasets 33
7.2 RPC vs PWI for emotions dataset 34
7.3 Experimental results for emotions dataset 34
7.4 RPC vs PWI for yeast dataset 35
7.5 Experimental results for yeast dataset 35
7.6 RPC vs PWI for scene dataset 35
7.7 Experimental results for scene dataset 36
7.8 RPC vs PWI for genbase dataset 36
7.9 Experimental results for genbase dataset 36
7.10 RPC vs PWI for medical dataset 37
7.11 Experimental results for medical dataset 37
7.12 Confusion matrix (Label 1, Label 2) emotions 38
7.13 Confusion matrix (Label 1, Label 2) yeast 40
7.14 Confusion matrix (Label 11, Label 12) yeast 40
7.15 Confusion matrix (Label 9, Label 11) genbase 41
7.16 Confusion matrix (Label 5, Label 6) genbase 41

1 Introduction

1.1 Motivation

Due to recent and ongoing developments in hardware, computers are becoming increasingly smaller and more powerful. This has also resulted in an ever growing storage density of devices, allowing many industries to acquire a wealth of data which requires algorithms for automated classification. From medical diagnostic technologies spotting tumors, to social networking sites searching for the latest trends, data classification is a very widespread topic. Classifiers need to run algorithms for solving a subset of a variety of problems. The problem we will be focusing on in this thesis is multilabel classification, which involves assigning one or more properties, called labels, to a data sample. The pursuit for high performance classifiers has lead to the development of sophisticated approaches such as various pairwise decomposition methods, which reduce complicated problems into smaller ones that are easier to solve. When learning classification models, many of these methods treat samples belonging to multiple labels as outliers. However, there is potential in classification algorithms which utilize information about label intersections. The goal of our work is to create such an algorithm and draw conclusions from its performance.

1.2 Multiclass Classification

Multiclass classification is a problem that involves assigning a single label λ from a set of disjoint labels L with $|L| > 2$ to a given sample instance. To achieve this, one must be able to make accurate assumptions based on a given sample's attribute values. Assigning labels to new samples, usually referred to as label prediction, is done by classifiers who were previously trained to solve the given problem type. The training process usually involves the classifiers consuming samples with known labels. Their algorithms attempt to extract as much information about the classes as possible and use it to build a classification model. After the model is built, the classifier is used to predict the labels of new and unknown samples. Since the assignment of a particular label can be seen as belonging to a specific class of data, the terms 'label' and 'class' are often used interchangeably.

1.3 Multilabel Case

The multilabel problem is an extension to multiclass classification. Here it is possible that an instance belongs to more than one label λ. This means that some classes overlap in feature space as seen in **figure 1.1**. Instances which belong to the intersecting area $A \cap B$ are often harder to correctly classify than those belonging to only one class.

The goal is to find the set of all relevant labels $Y \subseteq L$ for a given sample. Compared with the multiclass case, the multilabel problem is harder to evaluate, because the prediction for multilabeled samples includes more data. For example, when samples may belong to multiple classes,

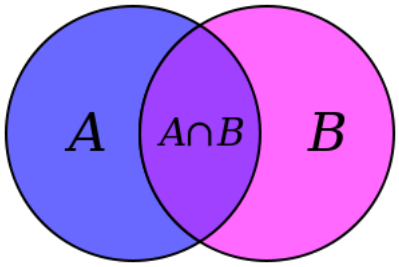


Figure 1.1: Overlapping classes (Wikipedia, 2015)

it is important to consider not only how many but also which relevant labels were and were not predicted. A popular way of evaluating multilabel classifiers is analyzing how they would order all labels, ranked from most to least relevant and with a threshold that draws the line between relevant and irrelevant. This is usually not a problem, since many multilabel classification algorithms produce label rankings as a side effect.

1.4 Outline

Section 2 will deal with formal definitions and baseline algorithms for multilabel classification, which are typically found in research papers. These will also be included in our own evaluations.
In **section 3**, we will give a short summary of related work that is relevant to this thesis.
In **section 4** and **section 5**, we will explain our approaches for solving the multilabel classification problem. This includes the training algorithms, the base classifiers and voting systems that were used.
Section 6 will be dedicated to our implementation and analysis of confusion matrices regarding multilabel classification. Potential weaknesses that can be revealed in classification approaches will also be discussed here.
In **section 7**, we will explain our experimental and evaluation processes, as well as present concrete results in a variety of settings.
Last but not least, in **section 8** we will summarize our discoveries and present ideas for future work.

2 Multilabel Classification

2.1 Definitions

Multilabel classification refers to the task of learning a function that maps instances $x \in X$ to subsets of class labels $P_x \subseteq L$, where $L = \{\lambda_1, ..., \lambda_c\}$ is a finite set of predefined labels. X contains data that is part of a particular concept. For example, in a concrete problem, instances $x \in X$ might represent music albums as a conjunction of attributes such as 'genre' and 'year', with labels such as 'jazz' and 'rock'. In contrast to multiclass learning, a single instance may belong to more than one label. We refer to P_x as the set of relevant labels for instance x. The set $N_x = L \setminus P_x$ are the irrelevant labels. Although the primary goal of multilabel classification is to find P_x for new data samples, other related problems exist. One of them is label ranking. Although it can be a separate task, it is also used for evaluating the performance of classifiers, whose main goal is only finding relevant labels. For better understanding, we are going to introduce some terms:

Label Cardinality of a multilabel dataset is the average number of labels of its samples.

Label Density of a multilabel dataset is the label cardinality divided by the number of labels in the dataset.

Label Ranking
Label ranking is a task which involves mapping instances x from an instance space X to rankings $\succ_x$ (total strict orders) over a finite set of labels $L = \{\lambda_1, ..., \lambda_c\}$, where $\lambda_i \succ_x \lambda_j$ means that, for instance x, label λ_i is preferred to λ_j. A ranking over L can be represented by a permutation as there exists a unique permutation τ such that $\lambda_i \succ_x \lambda_j$ if and only if $\tau(\lambda_i) < \tau(\lambda_j)$, where $\tau(\lambda_i)$ denotes the position of the label λ_i in the ranking. (Fürnkranz et al., 2008)

Classifiers
A classifier is a machine learning construct which attempts to learn information about instances and their label values from the training data it receives. We refer to this as the training process. The knowledge gained through training is used to build a classification model, which is ultimately used to solve the multilabel problem for new, unlabeled data. In other words, such a classifier creates a function $f : X \to P_x$. However, many classifiers used for multilabel learning can produce functions such as $f : X \to (Y \times \mathbb{R})^*$, which return pairs of labels with corresponding probabilistic estimations of the instance belonging to that label. We refer to the probabilistic values as 'confidence values'. When describing specific classifiers, it is important to state their training procedure. Throughout this thesis, we will use the notation λ_1 vs λ_2 to describe a classifier which distinguishes samples belonging to label λ_1 from those belonging to label λ_2. This implies that the classifier was trained with instances belonging to $\lambda_1 \cup \lambda_2$.

Overfitting
A classification model overfits the training data when it makes exaggerated assumptions about labels during the training procedure. This is usually caused by too few samples in the training set and leads to poor performance on test datasets. For example, if a classifier is trained to distinguish apples from other objects but its training dataset contains only a green apple, it might falsely learn to reject red apples.

2.2 Baseline Approaches

Most methods for solving the multilabel problem involve transforming it into multiple binary subproblems. This is done because binary classification problems are generally easier to solve, since they only need to determine one decision boundary. Binary classification has also been studied more extensively, which has lead to decent progress and results. The following chapter describes the most widespread standard approaches, which serve as a baseline for more complicated methods.

2.2.1 Binary Relevance

In binary relevance, aka. one vs. rest strategy, the classification problem is solved through training one classifier C_λ per existing label $\lambda \in L$. It is a method originally created for multiclass classification. Each C_λ creates a function $f_\lambda : X \to \{0, 1\}$, where the output 1 means the instance belongs to λ and 0 means it does not. In other words, for each class λ, a classifier is trained to build the model λ vs $L \setminus \lambda$. Instead of $\{0, 1\}$, the output can be different such as $\{-1, 1\}$ or a probabilistic value such as $[0, 1]$. For multiclass classification, the label λ is predicted, whose classifier C_λ returned a positive output. As required by the problem definition, only one class is predicted, even if multiple classifiers predicted that the instance belongs to their label.
Extended Binary Relevance implementations for the multilabel problem usually return a set of labels as the prediction, for some or all of their classifiers C_λ which returned a positive result.

2.2.2 Pairwise Decomposition

Multiclass
Classification through pairwise decomposition, aka. 'one vs. one strategy', involves dividing the multilabel problem into

$$\frac{n * (n-1)}{2}$$

binary subproblems, where n is the number of labels contained in the dataset.
For each pair of labels, one binary classifier is trained to distinguish between them. Multilabeled samples are considered as outliers and excluded from the training process. More strictly, for each pair of classes A and B, a classifier $C_{A,B}$ is trained on $A \setminus B$ vs $B \setminus A$. The outputs of each of them are aggregated with the help of a voting scheme to achieve the end prediction. Classifiers that are part of a decomposition method usually use a common algorithm, which is used for training and building the classification models. We refer to them as **base classifiers**.

Multilabel Variant
Pairwise classification for the multilabel case is mostly similar to the method for the multiclass problem. The main difference lies in the way the intersection areas between classes are handled. It is possible to apply the multiclass method to the multilabel case. Simply by removing the multilabeled samples from the training dataset, one can train classifiers exactly like in the multiclass problem, and thereafter use them for multilabel classification. However, treating multilabeled samples as outliers during the training phase is clearly not an optimal solution. Class intersections contain valuable information about multilabeled samples. Ideally, we want knowledge about the structure of overlapping areas to be contained in our classifiers' logic. In **section 3**, some efficient multilabel algorithms utilizing class intersections will be summarized.

2.2.3 Support Vector Machines (SVM)

Support vector machines are binary classification models which separate classes from each other by the maximum margin in feature space. The way they work can be visualized geometrically. First of all, a high-dimensional feature space suitable for the problem is determined. This is usually done in order to make classes linearly separable in space. After that, the training samples, based on their attribute values, are visualized as points in the space. The goal of an SVM is to construct a hyperplane which separates the two classes of samples by the highest margin. Before this, two hyperplanes that completely isolate both classes are defined. The samples which lie on these lines are called support vectors. The hyperplane which has the greatest distance to both of them is the optimal one. Given the proper feature space and training vectors

$$(y_1, x_1), ..., (y_l, x_l), \qquad y_i \in \{-1, 1\}$$

The optimal hyperplane is

$$w_0 \cdot x + b_0 = 0$$

which results from the following inequalities which describe the location of positive and negative samples:

$$\begin{aligned} w \cdot x_i + b \geq 1 \quad & if \quad y_i = 1 \\ w \cdot x_i + b \leq -1 \quad & if \quad y_i = -1 \end{aligned} \tag{2.1}$$

as well as the fact, that the optimal hyperplane lies where the distance $\rho(w, b)$ to both hyperplanes built from support vectors is the largest:

$$\rho(w, b) = \min_{\{x:y=1\}} \frac{x \cdot w}{|w|} - \max_{\{x:y=-1\}} \frac{x \cdot w}{|w|} \tag{2.2}$$

From (2.1) and (2.2) follows:

$$\rho(w_0, b_0) = \frac{2}{|w_0|} = \frac{2}{\sqrt{w_0 \cdot w_0}}$$

which is the distance between both dotted lines in **figure 2.1**.
The dotted lines are the hyperplanes which completely isolate samples belonging to one of the classes from all other samples. The blue line which lies in the middle of them is the optimal hyperplane which separates Class 1 from Class 2 by the highest margin.
The classification score that SVMs produce for a given sample is based on its distance to the separating hyperplane. Samples lying near the hyperplane are the most ambiguous instances that are the hardest to classify, while the ones lying furthest are the most 'obvious' representatives of one of the classes. The outputs of SVMs can be mapped to probabilities using Platt scaling (Platt et al., 1999).

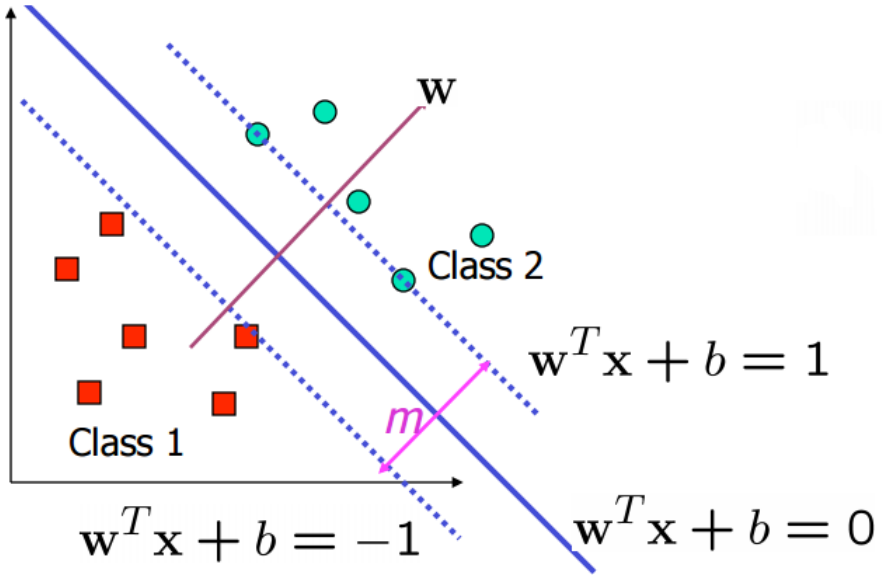


Figure 2.1: Support Vector Machine (Law, 2011)

2.2.4 Label Powerset

In the Label Powerset method, each combination of labels is considered a metaclass, for which a new classifier is trained. This leads to a large number of 2^L total classifiers and high computational complexity. Another drawback is the usually limited amount of instances for particular combinations of classes. If these are not handled separately, it could lead to overfitting. There exist optimized variants of Label Powerset, such as RAndom k-labELsets (RAkEL) (Tsoumakas and Vlahavas, 2007), which sets the maximum number of original classes that can build a metaclass to a number $k < L$.

2.2.5 Decision trees

Decision trees, in the context of multilabel classification, are models that can be represented as graphs that resemble trees. In order for an instance to be classified, its attributes must be tested; starting at the root node of the tree, moving down a certain number of branches and nodes, and ending at a leaf which contains the class value. Each node on the path, including the root, contains an attribute which needs to be inspected in the instance. Depending on its value, a certain branch is taken to the next node, and this is done recursively until a leaf is reached.

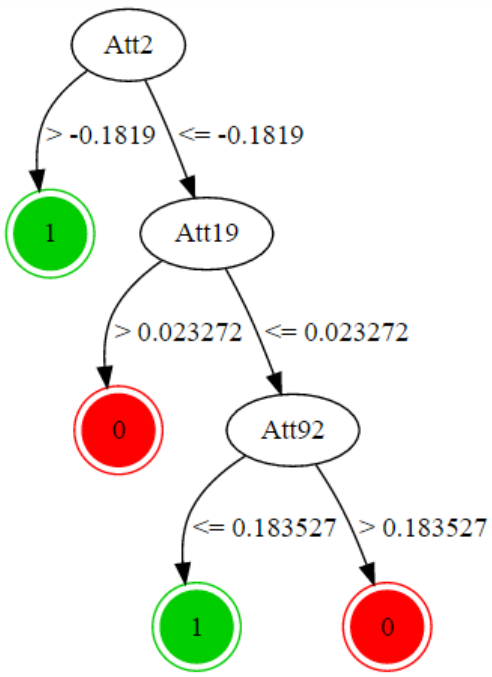


Figure 2.2: Decision tree

In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances (Mitchell, 1997). **Figure 2.2** shows an example decision tree, taken from our experiments on the dataset 'yeast'. The tree distinguishes samples belonging to only label 1 or both labels 1 and 13 (leaf value 1), from samples belonging to only label 13 (leaf value 0). The classifier was trained only on samples belonging to at least one of these labels.

For example, the instance with (Att2 $= -1$; Att19 $= 0$; Att92 $= 1$) would be sorted down the rightmost branch and would therefore be classified as a negative instance, which means it belongs only to label 13.
The classification of the instance with (Att2 $= 0$; Att19 $= x$; Att92 $= y$) would lead directly from the root to the leftmost leaf and therefore be predicted as a positive instance which belongs to only label 1 or both labels 1 and 13.

2.2.6 Voting systems

Voting systems determine the way that votes from multiple classifiers are combined to predict the labels of an instance. We distinguish two main types of voting:

1) **Binary voting**, where a classifier can only return, or whose outputs are interpreted as, binary values such as 0 and 1. It is used in basic approaches such as pairwise decomposition.

2) **Weighted voting**, where the outputs of the classifiers return values of varying confidence. The confidence value may be a value between 0 and 1.

3 Related Work

In **Paired Comparisons Method for Solving Multi-label Learning Problem** (Petrovskiy, 2006) a modified version of the traditional pairwise decomposition method, Multi-Label Paired Comparison (ML-PC) is presented. Instead of a single binary classifier, two are trained per pair of classes k and j. Let X be the input domain and Y the set of all possible classes. The number of different classes is $q = |Y| \geq 2$. S is a multilabel training set of size m:

$$S = \{(x_i, y_i) \mid 1 \leq i \leq m, x_i \in X, y_i \in Y\},$$

where each sample x_i is associated with a subset of relevant classes $y_i \subset Y$.

$F_S : X \rightarrow 2^q$ is the decision function that for any given sample x determines all its relevant classes. The training algorithm consumes samples from S labeled by class j, by k, or by both labels simultaneously. The first trained classifier estimates the following probabilities for class k:

$$r^+_{kj}(x) = P(k \in f_S(x) \wedge j \notin f_S(x) \mid x \in k \cup j), \tag{3.1}$$

$$r^-_{kj}(x) = 1 - r^+_{kj}(x) = P(j \in f_S(x) \mid x \in k \cup j) \tag{3.2}$$

analogously, the second classifier estimates the following probabilities for class j:

$$r^+_{jk}(x) = P(j \in f_S(x) \wedge k \notin f_S(x) \mid x \in k \cup j), \tag{3.3}$$

$$r^-_{jk}(x) = 1 - r^+_{jk}(x) = P(k \in f_S(x) \mid x \in k \cup j) \tag{3.4}$$

This results in the division of each pair of overlapping classes into four areas: overlapping, no one's area, only k and only j. This means that as opposed to the normal pairwise decomposition algorithm, which trains by separating single labeled instances, information about overlapping areas is gained during the training process of the classifiers. Probabilities, that a sample belongs to a particular area are calculated and based on these, thresholds for relevant classes are determined. This is solved by constructing an extended Bradley-Terry model, which was originally a model made for analyzing sports competitions. The algorithm used for fitting the calculated probabilities into the model is called minorization-maximization (MM). Since MM is computationally costly, Petrovskiy uses a simple filtering process for very large datasets. Based on the following voting scheme, one can assume that some percentage of the lowest ranked classes have a zero probability, and compute MM only on the rest of the classes.

$$vote_k(x) = \mid \{j \mid j \neq k \wedge r^-_{jk}(x) > 0.5\} \mid$$

This is shown to significantly improve classification speed at the cost of very slightly reduced precision.

An Improved Multi-label Classification Method Based on SVM with Delicate Decision Boundary (Chen et al., 2010) is a method which combines both Binary Relevance and Pairwise Decomposition using SVMs. It starts off with building one vs. rest models for each label. After that, one vs. one SVM surfaces for each pair of labels are created. These will be used to create pairwise bias models for certain pairs of labels. The reason behind them is that with help of threshold values from separating surfaces of both one vs. one models, as well as double-labeled instances, one can sometimes estimate the overlapping area of two labels. The bias models are created by estimating this area and then using it to train two areas which build the outer part of the overlapping area, the 'delicate decision boundaries'. This is done by selecting double labeled instances near to the SVM hyperplanes and estimating four range thresholds relative to both surfaces. The bias models are ultimately used to correct the results of those instances which lie in the area determined by the threshold values, which means they are located in overlapping areas.

A Multi-label Classification Algorithm Based on Triple Class Support Vector Machine (Wan and Xu, 2007)
In this paper, a variant of SVM which separates positive, negative and double label samples by using two parallel hyperplanes is implemented. A positive aspect of parallel SVMs is that, in contrast with nonparallel ones, they obey the closed world assumption. This means that the pairwise classifiers can never both output negative scores which means that a test sample will always be assigned to at least one of the classes. The classification algorithm starts off through pairwise decomposition. For each pair of classes, there are now four cases to consider: positive class vs. negative class, positive vs. mixed class, mixed class vs. negative class, and positive class vs. mixed class vs. negative class. Since the first three cases can be solved through typical binary SVM, the triple class SVM is only used to deal with the fourth case. The voting system is a binary one. A class receives one vote if the output of the SVM classifier crosses a certain threshold. If the triple class SVM predicts the mixed class, both classes will receive a vote.

In **Parallel and Sequential Support Vector Machines for Multi-label Classification** (Wang et al., 2005), the authors propose two algorithms: Parallel Support Vector Machines (PSVM) and Sequential Support Vector Machines (SSVM). PSVM is an algorithm which works similarly to triple class SVM. SSVM consists of two major steps. The first step is to decide, whether the instance belongs to the intersection of two labels $A \cap B$. This is achieved by building an SVM which trains $A \cap \overline{B}$ as the positive class and $\overline{A} \cap B$ as the negative class. The instances which belong to $A \cap B$ are duplicated. One of them gets labeled positive, the other negative. The resulting SVM classifier is then used to identify the double labeled instances. These are the ones of which the SVM outputs are near zero. The exact values for this are determined by a threshold. The second step of the algorithm, for instances which do not belong to $A \cap B$, involves deciding whether the instance belongs to $A \cap \overline{B}$ or $\overline{A} \cap B$. Since this is a typical binary problem, it is solved by an SVM trained by samples from both sets.

4 Pairwise with Intersections (PWI)

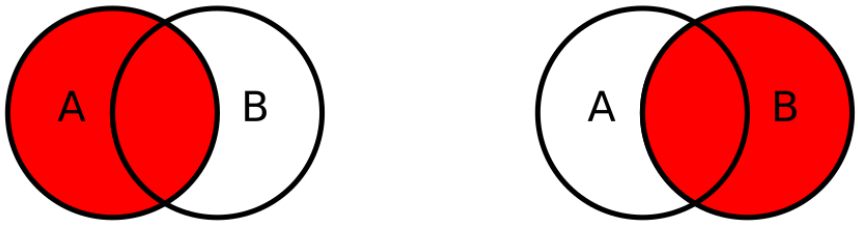
In this chapter, we will explain our method for solving multilabel classification. Our approach is a variant of pairwise classification for the multilabel problem and can be divided into two major steps: training and voting. We have implemented PWI in Java, and have used the open-source multilabel classification framework Mulan (Tsoumakas et al., 2011) for implementing as well as testing our method. PWI is inspired by the algorithm ML-PC (Petrovskiy, 2006). The training process of both algorithms is very similar. The voting stage of both is completely different, with ML-PC using the extended Bradley-Terry model and PWI using a system with manually set voting points.

4.1 Training

As shown in **section 2.2.2**, a typical pairwise decomposition approach trains one classifier per each pair of labels. In our work, for each pair of labels, we train two. This amounts to

$$n * (n-1)$$

different classifiers, where n is the number of labels the dataset contains.
These are trained to solve the subproblems A vs $B \setminus A$ as seen in **figure 4.1(a)** and B vs $A \setminus B$ as seen in **figure 4.1(b)**, for every pair of labels $A \neq B$. For each of these subproblems, only samples which belong to A, B or both are included in the training process. This implies that instead of treating the multilabeled samples as outliers which are to be removed before building the models, they are included in the training process. This lets information about the intersection area of labels be learned. Having two classifiers instead of one for each binary problem should also be beneficial in itself, since distributing two votes instead of one should reduce the classification error, provided both models are reasonably accurate.



(a) A vs $B \setminus A$ (b) B vs $A \setminus B$

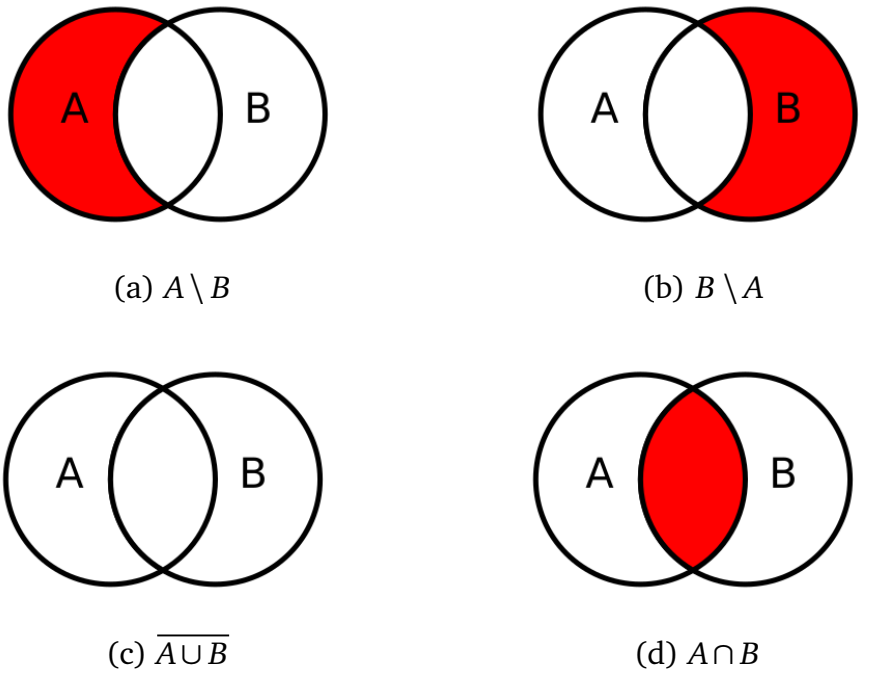
Figure 4.1: PWI training

4.2 Voting

This section will deal with PWI voting methods. In each voting system, voting points given by each classifier are aggregated.

4.2.1 Binary voting

Table 4.1 shows our binary voting settings. The first two columns of each voting system represent the outputs of the two classifiers A vs $B \setminus A$ and B vs $A \setminus B$ for each pair of labels A and B. The next two columns show the voting points both classes receive, based on these outputs.
When both classifiers return $-$, this means that the sample is predicted to belong to neither A nor B, as seen in **figure 4.2(c)**.
When A vs $B \setminus A$ returns $-$ and B vs $A \setminus B$ returns $+$, this means the sample should belong to only B, excluding the intersection A ∩ B. This case can be seen in **figure 4.2(b)**.



(a) $A \setminus B$ (b) $B \setminus A$

(c) $\overline{A \cup B}$ (d) $A \cap B$

Figure 4.2: PWI voting cases

When A vs $B \setminus A$ returns $+$ and B vs $A \setminus B$ returns $-$, this means the sample should belong to only A, excluding the intersection A ∩ B. This case can be seen in **figure 4.2(a)**.
When both classifiers return $+$, the prediction means that the sample lies specifically in A ∩ B, as seen in **figure 4.2(d)**.

In **Voting 0**, classes receive the same amount of voting points for double labeled samples as for single labeled ones. It is a straightforward voting system that is expected to show acceptable results.

Voting 1 is a variant where a class receives more points when the sample was classified as only belonging to one class. When comparing the results of this system to those of Voting 0, one should be able to see what impact the decreased importance of the double labeled samples has on the results.

In **Voting 2**, both classes receive a voting point, if neither classifier returns a $+$. These points

Voting 0				Voting 1				Voting 2			
A vs $B \setminus A$	B vs $A \setminus B$	A	B	A vs $B \setminus A$	B vs $A \setminus B$	A	B	A vs $B \setminus A$	B vs $A \setminus B$	A	B
$-$	$-$	0	0	$-$	$-$	0	0	$-$	$-$	1	1
$-$	$+$	0	1	$-$	$+$	0	2	$-$	$+$	0	2
$+$	$-$	1	0	$+$	$-$	2	0	$+$	$-$	2	0
$+$	$+$	1	1	$+$	$+$	1	1	$+$	$+$	2	2
Voting 3				Voting 4				Voting 5			
A vs $B \setminus A$	B vs $A \setminus B$	A	B	A vs $B \setminus A$	B vs $A \setminus B$	A	B	A vs $B \setminus A$	B vs $A \setminus B$	A	B
$-$	$-$	0	0	$-$	$-$	0	0	$-$	$-$	0	0
$-$	$+$	0	0	$-$	$+$	0	1	$-$	$+$	0	1
$+$	$-$	0	0	$+$	$-$	1	0	$+$	$-$	1	0
$+$	$+$	1	1	$+$	$+$	0	0	$+$	$+$	10	10

Table 4.1: PWI binary voting

are half as important as those from the other cases. Giving points to $-\,-$ doesn't seem to make much sense. We have used this voting scheme to see the impact of distributing points to classes that shouldn't receive them. It is expected to deliver slightly worse results than the previous two systems.

In **Voting 3**, classes only receive votes if the sample lies in the intersection area. This means that the typical cases of a sample belonging to just one label are ignored. It is a silly system which is expected to deliver terrible results.

Voting 4 is a typical voting scheme for pairwise classification. Samples that lie in the intersection area do not influence the output. It is expected that this voting system will produce good but not great results.

In **Voting 5**, the intersection area has the biggest influence on labeling the samples. It is an exaggerated method we used to see how much of an impact overrepresented double labeled samples have on our algorithm.

4.2.2 Weighted and hybrid voting

Weighted voting for PWI (**table 4.2**) requires that the base classifiers return confidence values. For this section, we will assume these values lie in $[0, 1]$. A simple and intuitive weighted voting scheme is **voting 6**, where in each case, the confidence values x_i provided by the base classifiers are used without further adjustments. This is similar to voting 0. Analogous to voting 1, in **voting 7**, single labeled predictions have a bigger impact on the resulting label ranking. In **voting 8**, we give more points to multilabel predictions than to single labeled ones. Through comparing the results of voting 6-8, one could note the performance of weighted voting in each pairwise classification case and use this knowledge to construct a hybrid voting system which combines binary and weighted voting. For example, if binary voting is proven to be best for the $+\,+$ case but weighted voting for the rest, then a good system could be something similar to

voting 9 (**table 4.3**). Likewise, if weighted voting is proven to be best for $+\,+$ but worse for the other cases, then a good system could look more like **voting 10**.

Voting 6				Voting 7				Voting 8			
A vs $B \setminus A$	B vs $A \setminus B$	A	B	A vs $B \setminus A$	B vs $A \setminus B$	A	B	A vs $B \setminus A$	B vs $A \setminus B$	A	B
$-$	$-$	0	0	$-$	$-$	0	0	$-$	$-$	0	0
$-$	$+$	0	x_B	$-$	$+$	0	$2 \cdot x_B$	$-$	$+$	0	x_B
$+$	$-$	x_A	0	$+$	$-$	$2 \cdot x_A$	0	$+$	$-$	x_A	0
$+$	$+$	x_A	x_B	$+$	$+$	x_A	x_B	$+$	$+$	$2 \cdot x_A$	$2 \cdot x_B$

Table 4.2: PWI weighted voting

Voting 9				Voting 10			
A vs $B \setminus A$	B vs $A \setminus B$	A	B	A vs $B \setminus A$	B vs $A \setminus B$	A	B
$-$	$-$	0	0	$-$	$-$	0	0
$-$	$+$	0	x_B	$-$	$+$	0	1
$+$	$-$	x_A	0	$+$	$-$	1	0
$+$	$+$	1	1	$+$	$+$	x_A	x_B

Table 4.3: PWI hybrid voting

5 Triple Class Pairwise (TCP)

Triple Class Pairwise (TCP) is our second method for solving multilabel classification. We have not implemented it for the purposes of this thesis due to time constraints. The main idea of using TCP is maximally utilizing the information about multilabeled samples, at the cost of reduced performance on specific datasets. The following description will be written in the same manner as **section 4**.

5.1 Training

TCP is a variant of the traditional pairwise decomposition approach for the multilabel problem, as described in **section 2.2.2**. We train

$$\frac{n*(n-1)}{2}$$

different classifiers, where n is the number of labels the dataset contains.
Each of these classifiers build a classification model $A \setminus B$ vs $A \cap B$ vs $B \setminus A$ for every pair of labels A and B with $A \neq B$. This means that the intersection area of two labels $A \cap B$ is seen as a new class during training. For each of these subproblems, only samples which belong to A, B or both are included in the training process. The three training areas can be seen in **figure 1.1**.

5.2 Voting

Table 5.1 shows voting systems for TCP. Under normal circumstances, which includes having proper training datasets, **voting 0** is expected to deliver good results. **Voting 1** and **voting 2** could be used to test the quality of the intersection predictions due to reasons stated in **section 5.3**. **Voting 3**, provided the base classifiers support confidence values, is a weighted voting alternative where x_i is the predicted confidence value for the label which 'won'. For example, when $A \setminus B$ is predicted, A receives an x_A amount of voting points, which is the exact confidence value for $A \setminus B$ returned by the classifier. The $B \setminus A$ case works analogous to $A \setminus B$. When $A \cap B$ is predicted, both A and B receive as many voting points as the confidence value for $A \cap B$.

Voting 0			Voting 1			Voting 2		
$A \setminus B$ vs $A \cap B$ vs $B \setminus A$	A	B	$A \setminus B$ vs $A \cap B$ vs $B \setminus A$	A	B	$A \setminus B$ vs $A \cap B$ vs $B \setminus A$	A	B
$A \setminus B$	1	0	$A \setminus B$	1	0	$A \setminus B$	1	0
$A \cap B$	1	1	$A \cap B$	0	0	$A \cap B$	10	10
$B \setminus A$	0	1	$B \setminus A$	0	1	$B \setminus A$	0	1

Table 5.1: TCP binary voting

Voting 3		
$A \setminus B$ vs $A \cap B$ vs $B \setminus A$	A	B
$A \setminus B$	x_A	0
$A \cap B$	x_A	x_B
$B \setminus A$	0	x_B

Table 5.2: TCP weighted voting

5.3 Outlook

The importance of the intersection areas grows with growing label cardinality and density in the testing dataset. Therefore, we expect the best results of TCP to be achieved on test datasets with high label cardinalities and densities, after having trained on datasets with sufficient samples with overlapping labels. A low label cardinality of the training datasets is a concern for TCP, since pairs of labels with very small overlapping areas may easily lead to overfitting. This is not that big of a problem for PWI, since in that approach, multilabeled samples are trained together with single labeled ones, so there usually is an acceptable sample size for each class during training. To avoid poor performance on datasets where there are too few training samples that belong to certain pairs of labels during training, one could take precautions for these specific labels. One could reduce the importance of the prediction, whenever our classifiers says the given test sample belongs to both of these labels, for example using weighted voting with a low value for this case. An other option could be abandoning the creation of the model $A \cap B$ altogether and switching to a traditional $A \setminus B$ vs $B \setminus A$ model.
In PWI, when both A vs $B \setminus A$ and B vs $A \setminus B$ returns $-$, we usually give neither A nor B a voting point. That is because the case $--$ implies that a given test sample belongs to neither of those classes. This case is not possible for TCP, since each case directly states that the sample belongs to either label A, B, or both. This means that some labels which are irrelevant will still receive some voting points. Under certain unfortunate circumstances, some irrelevant labels could be predicted as relevant even though they are completely irrelevant. The following example shows a potential problematic situation.

TCP trains on an average training set. Let $L = \{A, B, C..., Z\}$ be the set of labels in the given test dataset. Let $P_x = \{A\}$ be the set of relevant labels. Let the voting scheme be **voting 0**. Each classifier $A \setminus *$ vs $A \cap *$ vs $* \setminus A$ returns $A \setminus *$, which results in the label A receiving 25 voting points. However, each classifier $B \setminus *$ vs $B \cap *$ vs $* \setminus B$ returns $B \setminus *$, which results in label B receiving 24 points. Even though B is completely irrelevant, it was 'lucky' to win vs other irrelevant labels and achieve almost as many voting points as a relevant label.

This is not necessarily a problem, since irrelevant classes are unlikely to 'beat' other classes as consistently as in the example. Although this is an exaggerated example, the forced distribution of votes also affects less extreme settings. It is hard to say how big of an effect this may have without testing. Examining the difference between results of **voting 0** and **voting 3** would be a good start.

6 Confusion Matrix Analysis

The confusion matrix is a useful tool for analyzing the classification results of samples concerning specific one or two labels. The matrices we implemented show the prediction results of the pairwise base classifiers used for PWI. This means the values show the results of PWI before the aggregation and distribution of votes. Therefore, one must keep in mind that the matrices determine the quality of the unmodified base classifiers and not the whole classification algorithm PWI. Depending on the type of base classifiers, the values will be drastically different. Our data model contains one matrix per each pair of classes A and B, where $A \neq B$ and $A < B$. For the corresponding classes, each matrix shows the predictions of the classifiers A vs $B \setminus A$ and B vs $A \setminus B$, similar to the tables shown in Binary Voting in **section 4.2**.

Label A, Label B		Predicted class			
		$--$	$-+$	$+-$	$++$
Actual class	$--$	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
	$-+$	$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
	$+-$	$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
	$++$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

Table 6.1: Confusion matrix structure

The column headers of each table represent the number of instances that belong to the corresponding labels.

$--$ means the sample belongs to neither label A nor label B.
$-+$ means the sample belongs to label B but not to label A.
$+-$ means the sample belongs to label A but not to label B.
$++$ means the sample belongs to both label A and label B.

The rows of each table represent the true labels of the samples, while the columns show the labels predicted by the classifier. The diagonal represents the correctly classified samples. There are different ways for gaining knowledge about classifiers from a confusion matrix. We have implemented and used the following methods:

Row sensitivity
For a given matrix row, all entries are added up and divided by the value that belongs to the diagonal. The result is the percentage of correctly predicted samples which belong to the given 'actual class'. Using this metric, we can find out how well our classifier detects instances belonging to the given combination of two labels.

Column precision
For a given matrix column, all entries are added up and divided by the value that belongs to the

diagonal. The result is the percentage of the samples our classifier predicted to belong to the 'predicted class' which are actually correct. Formally, the result is the precision metric for the pair of labels shown in the column header.

Diagonal accuracy
The values that lie on the diagonal are summed up and divided by all values of the matrix. This gives us the accuracy with which the classifier predicts a sample that belongs to any combination of the two labels.

Class: Emotions					
Label 0, Label 1		Predicted class			
		− −	− +	+ −	+ +
Actual class	− −	28	147	48	87
	− +	9	51	20	30
	+ −	15	12	64	26
	+ +	6	12	14	24

Table 6.2: Confusion matrix example

The table above shows an example of a confusion matrix. It was calculated for label 0 and label 1 of the emotions dataset.
The first row of the matrix, $a_{0,x}$, contains all instances of the dataset with values label 0 = 0 and label 1 = 0. This means that emotions has $28 + 147 + 48 + 87 = 310$ instances for which label 0 = 0 and label 1 = 0.
The entry $a_{0,0} = 28$ means that 28 of the samples with actual class values label 0 = 0 and label 1 = 0 were predicted correctly.
147 of the label 0 = 0, label 1 = 0 samples were predicted as label 0 = 0, label 1 = 1.
48 of the label 0 = 0, label 1 = 0 samples were predicted as label 0 = 1, label 1 = 0.
87 of the label 0 = 0, label 1 = 0 samples were predicted as label 0 = 1, label 1 = 1.
The row sensitivity of $a_{0,x}$ is

$$\frac{28}{28 + 147 + 48 + 87} = 9\%$$

The column precision of the second column, $a_{x,1}$ is

$$\frac{51}{147 + 51 + 12 + 12} = 23\%$$

The diagonal accuracy of the matrix is

$$\frac{28 + 51 + 64 + 24}{593} = 28\%$$

6.1 Parent-child problem

When evaluating multilabel classifiers, it is interesting to see how they handle uncommon situations. The parent-child problem is a relatively rare case where a label λ_1 lies completely in label λ_2. In other words, λ_2 is a parent label of λ_1 when for every instance $x \in X$ is true that, if $\lambda_1 \in P_x$, then $\lambda_2 \in P_x$. This can often lead to misclassified samples. This situation is especially problematic for linear classifiers such as SVM, because it is not possible to separate these classes linearly in feature space.

6.2 Class pairs without intersections

Another case that is worth analyzing are classes with few or no intersections at all. These situations are equivalent to the typical scenario found in the multiclass problem. When attempting to solve a multilabel classification problem, usually the approach, such as pairwise classification, is applied to the whole problem. Due to this, some specific areas that deviate from the norm might not be solved optimally. For example, when solving the multilabel problem through pairwise decomposition with training on overlapping areas, there are likely to exist pairs of classes that do not have intersections at all or whose intersecting areas are very small. Confusion matrix analysis can also be used to analyze these special cases. It might be possible that for these classes, a different approach such as one for multiclass that ignores intersections would be best. This of course depends on the specific training strategies.

7 Experiments

7.1 Measures

The following measures will be used in our evaluations.

Average Precision
This metric calculates the average fraction of labels preceding relevant labels in the ranked list of labels.

$$Average\ Precision(P_x, \tau) = \frac{1}{|P_x|} \sum_{\lambda \in P_x} \frac{|\{\lambda' \in P_x | \tau(\lambda') \leq \tau(\lambda)\}|}{\tau(\lambda)} \quad \text{(Nam et al., 2014),}$$

where $\tau(\lambda)$ is the rank of label λ in the sorted list of labels.

Precision
This metric is defined as the fraction of samples predicted as relevant that actually are relevant.

$$Precision = \frac{tp}{tp + fp},$$

where tp is the number of true positives and fp the number of false positives.

Sensitivity
Sensitivity is defined as the fraction of samples belonging to a certain label, which were predicted as relevant.

$$Sensitivity = \frac{tp}{tp + fn},$$

where tp is the number of true positives and fn the number of false negatives.

IsError
This metric determines whether a given label ranking is perfect or not. For each sample, it returns 0 if all relevant labels of the given instance are correct and 1 otherwise.

RankingLoss
The RankingLoss metric computes the average fraction of pairs of labels which are not correctly ordered

$$RankingLoss(P_x, \tau) = \frac{|\ \{(\lambda, \lambda') \in P_x \times N_x \mid \tau(\lambda) > \tau(\lambda')\}}{|\ P_x\ ||\ N_x\ |} \quad \text{(Fürnkranz et al., 2008)}$$

Max F1
The Max F1 function uses the F1 metric, which combines recall and precision. Given is a label ranking τ as defined in **section 2.1**. The result of Max F1 for a test sample is the highest F1 value out of all possible F1 values.

$$MaxF1 = \max_{0 \leq k \leq K} \{F1_k\}$$

where $F1_k$ is the harmonic mean of recall and precision for the first k values of τ.

$$F1_k = 2 * \frac{recall_k * precision_k}{recall_k + precision_k},$$

where $recall$ is the number of predicted relevant labels, divided by the actual number of relevant labels. (Loza Mencia, 2006)

$$Recall = \frac{tp}{tp + fn},$$

where tp is the number of true positives and fn is the number of false negatives.

A given sample is a **true positive** when it passes a classification test as true, and its actual value is also true. In the context of our evaluations, true positives are labels that are classified as relevant for the given sample, and they are actually relevant.
A given sample is a **false positive** when it passes a classification test as true, but its actual value is false. In our context, false positives are labels that are classified as relevant for the given sample, but they are actually irrelevant.
A given sample is a **true negative** when it passes a classification test as false, and its actual value is also false. In our context, true negatives are labels that are classified as irrelevant for the given sample, and they are actually irrelevant.
A given sample is a **false negative** when it passes a classification test as false, but its actual value is true. In our context, false negatives are labels that are classified as irrelevant for the given sample, but they are actually relevant.

7.2 Base Classifiers J48

J48 is a classifier which generates decision trees for each classification problem. It is a Java implementation based on the C4.5 algorithm and part of the Weka software (Hall et al., 2009). The algorithm builds a decision tree based on the values of each attribute of the training samples. For each attribute, a tree node is created, so that samples can be discriminated most precisely. The criteria for creating such a node is highest information gain. **Figure 7.1** shows an example of a J48 decision tree model in text form. Each line contains a logical statement based on the values of attributes. Starting from the top, the statement is evaluated for the given sample. Depending on whether the line is a node or a leaf, a different action is taken. If the statement is true for the given sample and the current line is a leaf, then the prediction is made for the sample. The predicted label is given after the colon. The first number in the parenthesis

```
Mean_Acc1298_Mean_Mem40_Rolloff <= 0.117975
|   Std_Acc1298_Std_Mem40_MFCC_4 <= 0.130561: 1 (16.0/1.0)
|   Std_Acc1298_Std_Mem40_MFCC_4 > 0.130561
|   |   Mean_Acc1298_Std_Mem40_Flux <= 0.035437
|   |   |   Mean_Acc1298_Mean_Mem40_MFCC_6 <= 0.267363
|   |   |   |   Std_Acc1298_Std_Mem40_MFCC_1 <= 0.472392: 1 (3.0)
|   |   |   |   Std_Acc1298_Std_Mem40_MFCC_1 > 0.472392: 0 (4.0)
|   |   |   Mean_Acc1298_Mean_Mem40_MFCC_6 > 0.267363: 0 (28.0)
|   |   Mean_Acc1298_Std_Mem40_Flux > 0.035437: 1 (7.0/1.0)
Mean_Acc1298_Mean_Mem40_Rolloff > 0.117975
|   Mean_Acc1298_Mean_Mem40_MFCC_0 <= -85.230578
|   |   Std_Acc1298_Mean_Mem40_Flux <= 0.007916: 1 (4.0)
|   |   Std_Acc1298_Mean_Mem40_Flux > 0.007916: 0 (4.0)
|   Mean_Acc1298_Mean_Mem40_MFCC_0 > -85.230578: 1 (120.0/1.0)
```

Figure 7.1: J48 model

represents the number of training instances which reached this leaf. The second number is the number of those instances that are misclassified. If the statement is true and the current line is a node, then the line directly underneath is processed.
If the statement is false for the given sample and the current line is a node, then the line following the vertical dashed line is processed. If the current line is a leaf, the line underneath is processed.

7.3 Datasets

For our evaluations, we have used a subset of the multilabel datasets which are available on the Mulan website (Tsoumakas et al., 2011). The following table contains the most important information.

name	domain	instances	features	label cardinality	label density	labels
emotions	music	593	72	1.869	0.311	6
yeast	biology	2417	103	4.237	0.303	14
genbase	biology	662	1186	1.252	0.046	27
medical	text	978	1449	1.245	0.028	45
scene	image	2407	294	1.074	0.179	6

Table 7.1: Datasets

7.4 Evaluation

When testing and comparing classifiers, it is important to keep in mind that the difference in results between two or more classifiers partially depend on the given sample. Many algorithms, despite having different quality, might deliver very similar results when tested on particular data. Especially on smaller datasets, it is often not possible to distinguish a good algorithm from a poor one. For example, a classifier that can deal particularly well with label intersections could perform relatively poorly on a dataset with few overlapping classes. Analogously, a classifier that does well on a typical dataset might perform poorly if the dataset contains many multilabeled samples. In any case, it is crucial to have a big enough sample size when determining the general

performance of a classifier. Therefore, we have run our evaluations on multiple datasets, some of which have over 1000 samples.
The baseline algorithms we have used for testing are Java implementations which are part of the Mulan framework. Our algorithm was tested against Binary Relevance (BR) (Robert Friberg, 2012), Ranking by Pairwise Comparisons (RPC) (Hüllermeier et al., 2008), and Calibrated Label Ranking (CLR) (Fürnkranz et al., 2008).

7.4.1 Pairwise with Intersections

This section contains the tests which compare the general performance of PWI with the baseline algorithms. We have tested all 6 binary voting systems for PWI. All experiments have been run using 10-fold cross validation on the training set. The best score for a given metric is highlighted in yellow. In each evaluation, we also compare RPC to PWI regarding the training process and how it relates to their performance on the given dataset. 'Pairwise models' shows the number of models built during training by both algorithms. 'Training instances per model' is the number of unique instances used for training a single model. The range is built by the lowest and highest number of instances used for all models. The value in the parenthesis is the average number of instances used for training. The average is calculated over the results of all folds. Number of labels and label cardinality are the same values as seen in **table 7.1**.

Emotions dataset

algorithm	pairwise models	training instances per model	label cardinality	labels
RPC	15	93-403 (258.6)	1.869	6
PWI	30	184-410 (295.5)		

Table 7.2: RPC vs PWI for emotions dataset

Algorithm	Average Precision	IsError	Ranking Loss	Max F1
BR	0.7720 ± 0.0447	0.5500 ± 0.0830	0.1872 ± 0.0327	0.8164 ± 0.0358
RPC	0.7014 ± 0.0316	0.6595 ± 0.0540	0.2915 ± 0.0326	0.7561 ± 0.0211
CLR	0.7767 ± 0.0398	0.5600 ± 0.0663	0.1784 ± 0.0282	0.8256 ± 0.0291
PWIv0	0.7735 ± 0.0346	0.5363 ± 0.0669	0.1785 ± 0.0275	0.8291 ± 0.0275
PWIv1	0.7752 ± 0.0309	0.5312 ± 0.0669	0.1775 ± 0.0266	0.8281 ± 0.0221
PWIv2	0.7760 ± 0.0309	0.5279 ± 0.0592	0.1778 ± 0.0251	0.8335 ± 0.0236
PWIv3	0.6030 ± 0.0430	0.7808 ± 0.0530	0.3810 ± 0.0446	0.6935 ± 0.0374
PWIv4	0.7680 ± 0.0272	0.5482 ± 0.0576	0.1871 ± 0.0275	0.8159 ± 0.0234
PWIv5	0.6773 ± 0.0425	0.6879 ± 0.0606	0.2738 ± 0.0440	0.7567 ± 0.0381

Table 7.3: Experimental results for emotions dataset

The best results for the emotions dataset were delivered by PWI with **voting 2**, which had the best values for the IsError and Max F1 metrics. The other 'serious' classifiers are not far behind, except for RPC, which has relatively bad results and loses in each category. Its IsError is 0.1316 worse than the best score. Its average precision loses 0.0753 to the top score. The

difference in scores between RPC and PWI could be due to the fact that RPC excludes double labeled instances from the training process. In **table 7.2** we can see that PWI trained on 36.9 more instances on average than RPC. PWIv3 and PWIv5, as expected, delivered bad results.

Yeast dataset

algorithm	pairwise models	training instances per model	label cardinality	labels
RPC	91	13-1637 (929.9)	4.237	14
PWI	182	183-1930 (1123)		

Table 7.4: RPC vs PWI for yeast dataset

Algorithm	Average Precision	IsError	Ranking Loss	Max F1
BR	0.7401 ± 0.0220	0.7944 ± 0.0347	0.1823 ± 0.0136	0.7708 ± 0.0134
RPC	0.6216 ± 0.0151	0.9110 ± 0.0160	0.3097 ± 0.0115	0.6954 ± 0.0102
CLR	0.7459 ± 0.0214	0.7757 ± 0.0327	0.1783 ± 0.0131	0.7752 ± 0.0135
PWIv0	0.7449 ± 0.0232	0.7733 ± 0.0291	0.1786 ± 0.0142	0.7781 ± 0.0171
PWIv1	0.7524 ± 0.0236	0.7600 ± 0.0290	0.1728 ± 0.0147	0.7817 ± 0.0173
PWIv2	0.7486 ± 0.0228	0.7592 ± 0.0328	0.1742 ± 0.0142	0.7802 ± 0.0162
PWIv3	0.5207 ± 0.0136	0.9818 ± 0.0118	0.3606 ± 0.0140	0.6252 ± 0.0117
PWIv4	0.7434 ± 0.0221	0.7820 ± 0.0330	0.1811 ± 0.0138	0.7724 ± 0.0156
PWIv5	0.5574 ± 0.0145	0.9627 ± 0.0144	0.3127 ± 0.0128	0.6607 ± 0.0133

Table 7.5: Experimental results for yeast dataset

PWIv1 delivered the best results in this experiment, winning in every category except IsError. Most of the other classifiers come close to these scores. However, the gap between the performance of PWI and RPC is even bigger than the one in the emotions experiment. This is consistent with the difference in the number of training instances, which here is 193.1 on average. Furthermore, the minimal amount of training instances RPC used was as low as 13, while the lowest for PWI was 183. This suggests that for some specific label pairs, RPC produced very poor models, which suffered from issues such as overfitting. The high label cardinality of the dataset, 4.237, has a huge influence on these values.

Scene dataset

algorithm	pairwise models	training instances per model	label cardinality	labels
RPC	15	675-878 (678.6)	1.074	6
PWI	30	675-879 (764.8)		

Table 7.6: RPC vs PWI for scene dataset

Algorithm	Average Precision	IsError	Ranking Loss	Max F1
BR	0.7109 ± 0.0283	0.4450 ± 0.0431	0.2465 ± 0.0296	0.7716 ± 0.0206
RPC	0.7995 ± 0.0257	0.3631 ± 0.0414	0.1130 ± 0.0146	0.8580 ± 0.0189
CLR	0.8209 ± 0.0223	0.3249 ± 0.0363	0.1011 ± 0.0135	0.8728 ± 0.0144
PWIv0	0.7976 ± 0.0238	0.3656 ± 0.0390	0.1148 ± 0.0164	0.8598 ± 0.0177
PWIv1	0.8054 ± 0.0218	0.3519 ± 0.0356	0.1093 ± 0.0158	0.8650 ± 0.0163
PWIv2	0.8009 ± 0.0230	0.3611 ± 0.0399	0.1117 ± 0.0148	0.8633 ± 0.0176
PWIv3	0.4150 ± 0.0186	0.8401 ± 0.0214	0.5038 ± 0.0204	0.5307 ± 0.0159
PWIv4	0.8011 ± 0.0231	0.3594 ± 0.0357	0.1124 ± 0.0166	0.8626 ± 0.0179
PWIv5	0.6210 ± 0.0348	0.6012 ± 0.0507	0.2529 ± 0.0260	0.7198 ± 0.0250

Table 7.7: Experimental results for scene dataset

For the scene dataset, CLR produced the best results, winning in each metric. PWIv1 had the second best results. The gap between PWI and RPC has been lowered significantly. This is likely due to the fact that, even though PWI trained on 86.2 more instances on average, the minimal amount of training instances was equal for both algorithms. Also, the label cardinality is low for this dataset.

Genbase dataset

algorithm	pairwise models	training instances per model	label cardinality	labels
RPC	350-351 (350.9)	1-233 (53.9)	1.252	27
PWI	702	1-233 (54.6)		

Table 7.8: RPC vs PWI for genbase dataset

Algorithm	Average Precision	IsError	Ranking Loss	Max F1
BR	0.9927 ± 0.0042	0.0196 ± 0.0118	0.0028 ± 0.0036	0.9931 ± 0.0039
RPC	0.9914 ± 0.0056	0.0196 ± 0.0135	0.0083 ± 0.0058	0.9929 ± 0.0042
CLR	0.9914 ± 0.0056	0.0196 ± 0.0135	0.0083 ± 0.0058	0.9932 ± 0.0044
PWIv0	0.9908 ± 0.0049	0.0211 ± 0.0120	0.0083 ± 0.0055	0.9932 ± 0.0044
PWIv1	0.9908 ± 0.0049	0.0211 ± 0.0120	0.0083 ± 0.0056	0.9932 ± 0.0044
PWIv2	0.9908 ± 0.0049	0.0211 ± 0.0120	0.0083 ± 0.0056	0.9932 ± 0.0044
PWIv3	0.0944 ± 0.0269	0.9865 ± 0.0226	0.7309 ± 0.0378	0.2026 ± 0.0289
PWIv4	0.9906 ± 0.0050	0.0211 ± 0.0120	0.0084 ± 0.0056	0.9932 ± 0.0044
PWIv5	0.4512 ± 0.2079	0.8079 ± 0.2557	0.0894 ± 0.0572	0.5747 ± 0.1816

Table 7.9: Experimental results for genbase dataset

BR produced the best results for the genbase dataset, winning in 3 metrics. However, the score differences between all algorithms except PWIv3 and PWIv5 are all very small. Therefore, it is difficult to judge the quality of algorithms using this dataset.

Medical dataset

algorithm	pairwise models	training instances per model	label cardinality	labels
RPC	987-990(989.5)	1-346 (48.3)	1.245	45
PWI	1974-1980(1979)	1-346 (48.5)		

Table 7.10: RPC vs PWI for medical dataset

Algorithm	Average Precision	IsError	Ranking Loss	Max F1
BR	0.8341 ± 0.0278	0.2608 ± 0.0295	0.0743 ± 0.0193	0.8604 ± 0.0267
RPC	0.7534 ± 0.1959	0.3663 ± 0.2102	0.0604 ± 0.0609	0.7890 ± 0.1821
CLR	0.7606 ± 0.1873	0.3602 ± 0.2130	0.0503 ± 0.0433	0.7976 ± 0.1708
PWIv0	0.8733 ± 0.0194	0.2312 ± 0.0266	0.0269 ± 0.0091	0.8982 ± 0.0162
PWIv1	0.8724 ± 0.0181	0.2363 ± 0.0243	0.0269 ± 0.0092	0.8973 ± 0.0157
PWIv2	0.8724 ± 0.0190	0.2332 ± 0.0268	0.0270 ± 0.0090	0.8983 ± 0.0181
PWIv3	0.0554 ± 0.0076	1.0000 ± 0.0000	0.5907 ± 0.0430	0.1072 ± 0.0111
PWIv4	0.8710 ± 0.0157	0.2393 ± 0.0177	0.0270 ± 0.0092	0.8971 ± 0.0144
PWIv5	0.0929 ± 0.0138	1.0000 ± 0.0000	0.2660 ± 0.0411	0.1824 ± 0.0238

Table 7.11: Experimental results for medical dataset

In medical, the final dataset, PWIv0 delivers the best results. It wins in Average Precision, IsError, RankingLoss, and almost ties for the best Max F1 score. Despite having nearly the same average amount of training instances and the low label cardinality of the dataset, it beats RPC by a large margin.

7.4.2 Parent-child

We have also used our confusion matrix to see how classifiers deal with instances which are part of a parent-child constellation. In order to quickly find these cases for a given dataset, we ran a function isParentLabel(), which returns a boolean for every pair of labels in the dataset. We define a parent-child situation as a pair (Parent A, Child B) where A is a parent label of B as described in **section 6.1**. To visualize what is happening in a case (Parent A, Child B) or (Parent B, Child A), we can take a look at the confusion matrices for (Label A, Label B) and the corresponding classifier models for A vs $B \setminus A$ and B vs $A \setminus B$. We have run these tests on PWI using J48 base classifiers. Unless otherwise stated, classifier models for each pair of labels are taken from the a single random cross validation fold and they are very similar to models from the remaining folds.

Emotions

Since this dataset does not contain any parent-child constellations, we have temporarily modified it by manually deleting particular instances. After doing this, label 1 was a parent label of label 2 for the duration of this test.

We can see in the last row of **table 7.12** that the total number of double labeled instances for (Label 1, Label 2) is $25 + 66 = 91$.

The **row sensitivities**, from top to bottom are 0%, NaN, 69.3%, 72.5%.

The **column precisions**, from left to right, are NaN, NaN, 21.4%, 37.1%.
The **diagonal accuracy** is 28.1%.

```
J48 pruned tree
------------------

Std_Acc1298_Std_Mem40_MFCC_11 <= 0.079917
|   BH_LowPeakBPM <= 54: 1 (2.0)
|   BH_LowPeakBPM > 54: 0 (36.0/1.0)
Std_Acc1298_Std_Mem40_MFCC_11 > 0.079917
|   Mean_Acc1298_Mean_Mem40_MFCC_0 <= -65.629025
|   |   Mean_Acc1298_Mean_Mem40_MFCC_1 <= 2.667606: 0 (5.0)
|   |   Mean_Acc1298_Mean_Mem40_MFCC_1 > 2.667606
|   |   |   BHSUM2 <= 0.234702
|   |   |   |   Std_Acc1298_Std_Mem40_MFCC_4 <= 0.08753: 0 (3.0/1.0)
|   |   |   |   Std_Acc1298_Std_Mem40_MFCC_4 > 0.08753: 1 (46.0/1.0)
|   |   |   BHSUM2 > 0.234702
|   |   |   |   Std_Acc1298_Mean_Mem40_MFCC_8 <= 0.254794: 0 (9.0/1.0)
|   |   |   |   Std_Acc1298_Mean_Mem40_MFCC_8 > 0.254794
|   |   |   |   |   BHSUM1 <= 0.153613: 0 (3.0)
|   |   |   |   |   BHSUM1 > 0.153613
|   |   |   |   |   |   BH_LowPeakBPM <= 59: 0 (4.0/1.0)
|   |   |   |   |   |   BH_LowPeakBPM > 59: 1 (29.0/2.0)
|   Mean_Acc1298_Mean_Mem40_MFCC_0 > -65.629025: 0 (10.0)

Number of Leaves  :     10

Size of the tree :      19
```

Figure 7.2: J48 model label 2 vs label 1 \ label 2 emotions

Label 1, Label 2		Predicted class			
		− −	− +	+ −	+ +
Actual class	− −	0	0	165	89
	− +	0	0	0	0
	+ −	0	0	52	23
	+ +	0	0	25	66

Table 7.12: Confusion matrix (Label 1, Label 2) emotions

```
J48 pruned tree
------------------
: 1 (149.0)

Number of Leaves  :     1

Size of the tree :      1
```

Figure 7.3: J48 model label 1 vs label 2 \ label 1 emotions

The J48 model for label 1 vs label 2 \ label 1 (**figure 7.3**) simply assigns label 1 to every test instance, ignoring all attribute values. The label 2 vs label 1 \ label 2 model (**figure 7.2**) on the other hand is relatively complicated, with a tree size of 19. By adding up the values in the leaves of the tree, we can see that 77 of the 147 instances used for training this model reached a leaf with value 1. This means that these instances were built for paths leading to a label 2 prediction. The rest, which is slightly under half of the instances, reached a leaf with value 0. The equal distribution of both sides shows, to some extent, that test instances have a roughly

equal chance of receiving a label 2 or non-label 2 prediction.

Verdict emotions
Overall, we can say that the classification results for this dataset are relatively bad. Only 28.1% of instances have been classified perfectly by the (Label 1, Label 2) classifiers. Due to the simplistic J48 model for label 1 vs label 2 \ label 1, no single instance belonging to the − − case can be classified properly. The column precisions of 21.4% and 37.1% seem low.

Yeast
Through modifying the data as described in the previous test, we have achieved five parent-child label pairs. Since three of these cases had only three double labeled instances each, we will ignore them due to sample size issues and take a look at the remaining two pairs, (Parent 1, Child 2) and (Parent 11, Child 12), where label 11 is a parent label of label 12.
The total number of instances in the modified dataset is 1988.

The total number of double labeled instances for (Label 1, Label 2) is 554.
The **row sensitivities**, from top to bottom are 0%, NaN, 65.1%, 66.7%.
The **column precisions**, from left to right, are NaN, NaN, 32.7%, 36%.
The **diagonal accuracy** is 34%.
The J48 model for label 1 vs label 2 \ label 1 (**figure 7.4**) is similar to the one for the emotions dataset. Because of this, all − − are misclassified here as well. The J48 model for label 2 vs label 1 \ label 2 has a tree size of over 150, and therefore is not to be seen here.
The precision values also seem low, but they are better than those for the emotions dataset.

The total number of double labeled instances for (Label 11, Label 12) is 1456.
The **row sensitivities**, from top to bottom are 0%, NaN, 0%, $\sim$ 100%.
The **column precisions**, from left to right, are NaN, NaN, 0%, 73.3%.
The **diagonal accuracy** is 73.2%.
Both classifier models (**figure 7.5**) always return 1. There are 2 outlier instances which only received label 11 for an unknown reason. The models deliver decent precision for double labeled instances, which make up the most part of test instances. For every other case, instances will be misclassified. However, this is not a big problem, when most test instances are double labeled.

Verdict yeast
The third row of the confusion matrix (Label 11, Label 12) shows that label 11 is almost equal to label 12, since there are very few instances belonging to only label 11. This classifier pair delivers acceptable results. The biggest concern for both pairs of classifiers for the yeast dataset are the − − cases. Avoiding the labelling of − − instances would drastically increase performance.

Label 1, Label 2		Predicted class			
		− −	− +	+ −	+ +
Actual class	− −	0	0	463	487
	− +	0	0	0	0
	+ −	0	0	315	169
	+ +	0	0	185	369

Table 7.13: Confusion matrix (Label 1, Label 2) yeast

```
J48 pruned tree
------------------
: 1 (930.0)

Number of Leaves  :     1

Size of the tree :  1
```

Figure 7.4: J48 model label 1 vs label 2 \ label 1 yeast

Label 11, Label 12		Predicted class			
		− −	− +	+ −	+ +
Actual class	− −	0	0	1	515
	− +	0	0	0	0
	+ −	0	0	0	16
	+ +	0	0	1	1455

Table 7.14: Confusion matrix (Label 11, Label 12) yeast

```
J48 pruned tree
------------------
: 1 (1330.0)

Number of Leaves  :     1

Size of the tree :      1
```

(a) J48 model label 11 vs label 12 \ label 11 yeast

```
J48 pruned tree
------------------
: 1 (1319.0/16.0)

Number of Leaves  :     1

Size of the tree :      1
```

(b) J48 model label 12 vs label 11 \ label 12 yeast

Figure 7.5: J48 models for (label 11, label 12) yeast

Genbase

There was no need to modify this dataset, since it already contained 19 parent-child pairs. These pairs have from 2 to 29 different double labeled instances each. We will examine the cases with the most double labeled instances (Parent 9, Child 11) and (Parent 6, Child 5).
The total number of instances in the dataset is 662.

The total number of double labeled instances for (Label 9, Label 11) is 29.

The **row sensitivities**, from top to bottom are 0, NaN, 100%, 100%.
The **column precisions**, from left to right, are NaN, NaN, 5.9%, 85.3%.
The **diagonal accuracy** is 10%.
In this case, another classifier which always returns 1 is built (**figure 7.6(a)**). The label 11 vs label 9 \ label 11 classifier is a small tree of size 2, which bases its prediction on the value of a single attribute (**figure 7.6(b)**). The only good precision result is the one for + + cases, which involves relatively few instances. Like in the emotions and yeast datasets, − − cases are always classified wrongly. The fact that most of the test instances belong to neither of both labels makes this pair of classifiers deliver particularly terrible results.

The total number of double labeled instances for (Label 5, Label 6) is 23.
The **row sensitivities**, from top to bottom are 0%, 100%, NaN, 100%.
The **column precisions**, from left to right, are NaN, 1%, NaN, 100%.
The **diagonal accuracy** is 5%.
The classifiers for this case are similar to those from the previous case. This time, all + + instances are classified correctly, but the amount of − − makes this the worst classifier from this whole section.

Verdict genbase
Both genbase classifiers seem to be the worst out of this section, having only a good precision for the ++ case. Since this dataset has a low label cardinality of 1.252, the low precision for the − + and + − cases should have a particularly bad effect on classification results. Since the results for this dataset in the previous section are very decent (**table 7.9**), the classifiers for (Label 9, Label 11) and (Label 5, Label 6) are likely to be one of the worst of those built for this dataset.

Label 9, Label 11		Predicted class			
		− −	− +	+ −	+ +
Actual class	− −	0	0	591	5
	− +	0	0	0	0
	+ −	0	0	37	0
	+ +	0	0	0	29

Table 7.15: Confusion matrix (Label 9, Label 11) genbase

Label 5, Label 6		Predicted class			
		− −	− +	+ −	+ +
Actual class	− −	0	631	0	0
	− +	0	8	0	0
	+ −	0	0	0	0
	+ +	0	0	0	23

Table 7.16: Confusion matrix (Label 5, Label 6) genbase

```
J48 pruned tree
------------------
: 1 (60.0)

Number of Leaves  :     1

Size of the tree :  1
```

(a) J48 model label 9 vs label 11 \ label 9 genbase

```
J48 pruned tree
------------------

PS50002 = YES: 1 (27.0)
PS50002 = NO: 0 (34.0)

Number of Leaves  :     2

Size of the tree :  3
```

(b) J48 model label 11 vs label 9 \ label 11 genbase

Figure 7.6: J48 models for (label 9, label 11) genbase

```
J48 pruned tree
------------------

PS50004 = YES: 1 (20.0)
PS50004 = NO: 0 (8.0)

Number of Leaves  :     2

Size of the tree :  3
```

(a) J48 model label 5 vs label 6 \ label 5 genbase

```
J48 pruned tree
------------------
: 1 (28.0)

Number of Leaves  :     1

Size of the tree :  1
```

(b) J48 model label 6 vs label 5 \ label 6 genbase

Figure 7.7: J46 models for (label 6, label 5) genbase

Summary

J48 classifiers build very simple classification models for the classes of parent-child dependencies. Though the models may be good enough for typical applications, not a single — — instance was correctly classified in this section. This shows that it is not possible to obtain excellent results with J48 in parent-child settings.

8 Conclusion and Future Work

In this thesis, we have implemented Pairwise with Intersections, our variant of the pairwise decomposition approach for solving the multilabel classification problem. In our experiments, PWI has achieved better results than Ranking by Pairwise Comparisons, a typical algorithm which ignores multilabeled samples during the training process. This shows the importance of the information contained in the overlapping areas of classes. PWI showed better results than the other baseline algorithm, Binary Relevance. It can compete with more sophisticated algorithms such as Calibrated Label Ranking as well. We have also constructed a confusion matrix for the purpose of analyzing the performance of base classifiers. This has been used to examine how J48 classifiers deal with parent-child label pairs. Tests have revealed the poor performance of J48 in these settings, due to overly simplistic classification models. The experiments from **section 7.4.1** have shown some surprising results, especially the ones for the medical dataset, where large differences in performance have been observed. It could be interesting to look more into such scenarios and find the reasons behind the outcomes.

Future work should ideally involve testing PWI on more datasets, as well as using weighted and mixed voting. A different algorithm we proposed for datasets with high label cardinality, Triple Class Pairwise, has not been implemented yet. As mentioned in **section 6**, our confusion matrices deliver the results of the base classifiers and not the final results of PWI. Since results of decomposition algorithms consist of aggregations of sub results, the results of a single pair of base classifiers are usually much different than the combined results after aggregation. Therefore, in order to properly judge how good a specific base classifier performed for the given algorithm, one has to see the relationships between the pre and post voting results. It is possible to extend our confusion matrix data structure for it to display the classification results after the aggregation of all pairwise classifiers' scores. This data can then be used for tests similar to those in **section 7.4.2**. Through comparing the matrices of base classifiers with the matrices of PWI after voting, we could draw more conclusions about PWI and its base classifiers.

Bibliography

Benhui Chen, Liangpeng Ma, and Jinglu Hu. An improved multi-label classification method based on svm with delicate decision boundary. *International Journal of Innovative Computing, Information and Control*, 6(4):1605–1614, 2010.

Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Machine learning*, 73(2):133–153, 2008.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656278. URL http://doi.acm.org/10.1145/1656274.1656278.

Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Ranking by pairwise comparisons, 2008. URL http://sourceforge.net/p/mulan/gitcode/ci/e1f0d4f8e5c35af0ef158dac382ba1917588e15b/tree/mulan/src/mulan/classifier/transformation/Pairwise.java. Accessed: 2015-08-30.

Martin Law. A simple introduction to support vector machines, 2011. URL http://www.cise.ufl.edu/class/cis4930sp11dtm/notes/intro_svm_new.pdf. Accessed: 2015-08-30.

Eneldo Loza Mencia. Paarweises lernen von multilabel-klassifikationen mit dem perzeptron-algorithmus. Master's thesis, TU Darmstadt, Knowledge Engineering Group, March 2006. URL http://www.ke.tu-darmstadt.de/lehre/arbeiten/diplom/2006/Loza_Eneldo.pdf. Diplomarbeit.

Tom M Mitchell. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45, 1997.

Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. Large-scale multi-label text classification—revisiting neural networks. In *Machine Learning and Knowledge Discovery in Databases*, pages 437–452. Springer, 2014.

M. Petrovskiy. Paired comparisons method for solving multi-label learning problem. In *Hybrid Intelligent Systems, 2006. HIS '06. Sixth International Conference on*, pages 42–42, Dec 2006.

John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Grigorios Tsoumakas Robert Friberg. Binary relevance, 2012. URL http://mulan.sourceforge.net/doc/mulan/classifier/transformation/BinaryRelevance.html. Accessed: 2015-08-30.

Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Machine learning: ECML 2007*, pages 406–417. Springer, 2007.

Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.

Shu-Peng Wan and Jian-Hua Xu. A multi-label classification algorithm based on triple class support vector machine. In *Wavelet Analysis and Pattern Recognition, 2007. ICWAPR'07. International Conference on*, volume 4, pages 1447–1452. IEEE, 2007.

Liwei Wang, Ming Chang, and Jufu Feng. Parallel and sequential support vector machines for multi-label classification. *International Journal of Information Technology*, 11(9):11–18, 2005.

Wikipedia. overlapping classes a and b, 2015. URL `https://upload.wikimedia.org/wikipedia/commons/thumb/6/6d/Venn_A_intersect_B.svg/350px-Venn_A_intersect_B.svg.png`. Accessed: 2015-08-30.