LUDWIG-MAXIMILIANS-UNIVERSITÄT
TECHNISCHE UNIVERSITÄT MÜNCHEN

Institut für Informatik der Technischen Universität München;
Lehrstuhl für Bioinformatik

Bachelorarbeit
in Bioinformatik

Feature and Label-Selection for Multi-Label Classification

Sebastian Richard Lehnerer

Aufgabensteller:	Prof. Dr. Stefan Kramer
Betreuer:	Dipl.Bioinf. Jörg Wicker
Abgabedatum:	15.10.2011

Ich versichere, dass ich diese Diplomarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 14. Oktober 2011 Sebastian Lehnerer

Abstract

This bachelor thesis presents a new approach for Feature and Label Selection for Multi-Label Classification. In general, classification is the task of assigning one or more labels to an unseen instance. Multi-label classification is concerned with the assignment of a set of labels is assigned to each instance. This is useful for applications such as protein function annotation and prediction of toxicological endpoints. This work bases on the assumption that there exist subsets of the label and feature space having a higher internal dependence than to the rest of data. Here, two methods are proposed to identify those groups and apply any multi-label learning scheme on it. It can be shown that splitting the dataset into subgroups does not affect the performance, yet it does yield improvements in some cases.

Diese Bachelorarbeit zeigt einen neuen Ansatz zur Feature und Label-Auswahl zur Multi-Label Klassifizierung. Allgemein beschreibt Klassifizierung die Aufgabe unbekannten Objekten eine oder mehrere Klassen zuzuordnen. Dieser Schritt wird Multi-Label Klassifizierung genannt, wenn jedem Objekt eine Menge von Klassen zugeordnet wird. Dies ist unter anderem nützlich zur Protein-Funktionsanalyse sowie zur Bestimmung der Toxizität unklassifizierter Substanzen. Diese Arbeit basiert auf der Annahme, dass innerhalb dieser Daten Untergruppen von Features und Labels bestehen die einen größeren Zusammenhang aufweisen als zu den restlichen Attributen. Zwei Methoden wurden entwickelt um diese Untergruppen zu identifizieren und Multi-Label Klassifikatoren auf diesen zu lernen. Es konnte gezeigt werden, dass die Zerteilung der Feature- und Label-Menge keinen negativen Einfluss auf die Vorhersagegenauigkeit hat. In einigen Fällen konnte sogar eine Verbesserung festgestellt werden.

Contents

1 Introduction and Related Work

1.1 Description of the field

Computational biology has become an essential part of modern biology, biochemistry and medicine. Closely related fields like chemoinformatics, immunoinformatics and system biology spread the tools of computational science to many fields of life science. Bioinformatics or computational biology aims to collect, categorize, store and analyze biological systems [17]: Some time before the Human Genome Project [13] biologists became aware of the powerful possibilities of computational science. Successively a huge amount of data, e.g. gene sequences, structural information, interactions and other, data was collected. In the beginning, bioinformatics enabled the storage of those data in databases like SWISSPROT [2] and the Protein Data Bank (PDB)[1]. After storing and cataloging data, analyzing steps to discover patterns and structures within the data [17] were developed.

Knowledge discovery in databases (KDD) [9] processes large datasets and aims to discover patterns within the data using statistical methods as well methods from artificial intelligence. It offers deep insight into the data and a better understanding of the underlying patterns. The process of mining patterns describing the data is called descriptive data mining. On the other hand, predictive data mining allows extracting models and offers methods to classify unseen examples.

Unsupervised learning is the task to find patterns like clusters in unlabeled data. Supervised learning refers to the task of learning models to classify labeled data. It maps an instance of features to an output variable called label. Supervised learning can be split into classification and regression analysis. While classification predicts discrete values, regression analysis offers methods to predict continuous target variables. Besides clustering and association rule learning, classification is one of the core areas in data mining.

Classification is the process of assigning categories to examples of data [19]. A common example is the *PlayTennis*-Problem whether to play tennis or not. Given a dataset which maps features like weather or weekday to the decision

[1] http://www.rcsb.org/pdb/

weather	weekday	class
sunny	Sunday	yes
sunny	Monday	no
rain	Sunday	no

Table 1.1: Example for classification: Columns describe features, the last column represents the class. The class corresponds to the decision if the user plays tennis or not. Examples are showed in rows.
If the tennis court is only available on weekends and the user only plays tennis on sunny days only the first instance will be classified to yes, both other instance will be classified to no.

play tennis or not, the goal is to learn a model predicting this target value using the given features. An example is shown in table 1.1.

Amongst others, there are two basic extensions of the task of classification: 1) from a binary to a multi-class classification problem, where more than two categories are available and 2) to a multi-label problem. The latter one is given if an instance is associated with a set of categories (labels) [29].

Definition Formally, a dataset D can be described as follows: $D = \{(x_i, Y_i)|i = 1...|D|\}$, where each instance (x_i, Y_i) consists of a feature vector x_i and a subset of labels/classes $Y_i \subseteq L$, where L is the full set of labels. Single-label classification is concerned if a class $\lambda \in L$ is assigned to instances. Binary classification refers to label sets L of cardinality 2 ($|L| = 2$), multi-class classification to sets $|L| > 2$. In both cases classifiers choose out of $|L|$ classes but always only one class λ is assigned to an instance. In multi-label classification, a subset $Y \subseteq L$ is assigned to each instance, therefore a classifier has to choose a set of assigned labels as well as the number of assigned labels. Another aspect of multi-label classification is to produce a ranking of labels along to its relevance to the current example.

Applications For instance, pictures mostly match not only one category, but show a combination of different categories like hill, forest, water or beach. Another example would be movies like *Da Vinci Code* which can be categorized as Society\Religion and Arts\Movies [29].

An application in bioinformatics for multi-label classification is the ToxCast™ project [7]. The project started in 2007. 320 different chemical structures were tested for toxicological endpoints. About 1600 properties and more than

400 endpoints were examined.

The goal of this project is to map *in vitro* measured features to *in vivo* observed toxicological endpoints, which is a classic task of classification. Particularly, it is a challenging task for multi-label learning, as the set of toxicological endpoints (400), which refer to labels and the set of features (1600), is quite large. Once this is done, predictions about the toxicity of chemical substances/structures can be made without tedious lab-work or morally questionable animal experiments.

1.2 Motivation

Task. *For a dataset* $D = \{(x_i, Y_i)|i = 1...|D|\}$ *with a feature vector* x_i*, label set* $Y_i \subseteq L$*, there exists a subset* $L_G \subseteq L$ *of all labels and a corresponding subset* $X_G \subseteq X$ *of features forming a subset of the data* $G \subseteq D$ *with* $G = \{(x_{Gi}, Y_{Gi})|i = 1...|G|\}$*, in which attributes have a high internal association.*

In many multi-label applications there exist groups of connected labels and features which come naturally. E.g. in the picture labeling example, it is probable to see combinations of water, beach and island. On the other hand, it is very improbable to see combinations like forests and desert. Another example would be the ToxCast™ dataset, where it is highly probable to find families of in vitro features connected to a group of similar in vivo endpoints. Therefore, there will be specific features for specific labels. For instance, the blue content of a picture may be important for the label *sea* and *sky*, but irrelevant to other labels like *forests*.

Based on this assumption, this work aims to develop a method to find sets of labels and corresponding features. Subsequently, multi-label learners can be applied on those sub-datasets. Experiments show that those models achieve better performance in some cases since they can fit to characteristic groups. In addition, subsets representing independent information patterns can be isolated.

The bachelor thesis is organized as follows: Firstly core components like clustering and feature selection are described. Subsequently, two Feature- and Label-Selection for Multi-Label Classification methods are presented in this work. Finally, it concludes with a presentation of the evaluation and discussion of benefits of these methods.

1.3 Related Work

This chapter gives an overview on existing multi-label classification techniques. HOMER is described here, as it uses a similar approach to this work, despite its focus lies on efficiency rather than structural decomposition.

Notation The following notation is used for describing the multi-label problems: D is the dataset composed of n examples $(x_1, Y_1), (x_2, Y_2), \ldots, (x_n, Y_n)$, where x_i is the instance containing the feature values and $Y_i \subseteq L$ a subset of labels associated with this instance. A vector $\vec{y_i} = [y_{i_1} \ldots y_{i_L}] = \{0, 1\}^L$ is used to describe Y_i, where $\vec{y_{i_j}} = 1$ if the jth label is relevant (otherwise $\vec{y_{i_j}} = 0$). Single labels are annotated with $\lambda_{1 \ldots L}$.

1.3.1 Multi-Label Classification Methods

There are two groups of multi-label classification methods [29]:

1. Transformation-based methods, which transform the problem in to several single-label problems and apply binary classification methods (e.g. SVMs [3], Decision Trees [23], k-Nearest-Neighbors [19]). An exception is the Label Powerset method, which transforms the problem to a multi-class classification task.

2. Algorithm-adaptation methods, which extend existing algorithms to be capable of handling multi-label problems directly.

Transformation based Methods

Binary Relevance A basic multi-label classification method is Binary Relevance (BR) [29]. This method transforms the multi-label problem into $|L|$ separate problems and trains a single-label learner on each on them. The prediction is simply composed by the prediction of the single classifications $\lambda_1 \ldots \lambda_n$. It does not take any dependencies between labels into account. This makes it the simplest multi-label learner. Labels tend to have strong inter-relationships, however, BR does not consider those relationships, thus predictions on a label l_i have no influence on predictions of other labels l_j. However, it is still popular as it scales linear with the number of labels, thus complexity lies in $O(|L|)$.

Label Powerset The Label Powerset method includes label dependencies but suffers from computational complexity. For every unique set Y of labels in the data, a class l_Y is created, representing this combination. Subsequently, and a multi-class classifier is built on the data. Thus, every combination is represented as a single class and label correlations are taken into account directly. Despite the worst-time complexity of $O(2^{|L|})$ another disadvantage is that, in its basic form, it can only predict label combinations which are present in the training data.

Multi-Label Stacking Multi-label Stacking or BR^2 deals with the disadvantage of BR by not taking label relations into account [28]. Therefore, a stacking mechanism is used. The first level of Multi-label Stacking consists of a BR classifier predicting labels independently. A second level incorporates the output of the first level BR classifier, in detail the label confidences. Result is a relation-dependent prediction. Formally, first step produces a dataset $D' = \{(y_i, Y_i), i = 1 \dots |D|)\}$. The vector y_i contains all predicted confidences of the base level for all labels $\lambda_{1\dots|D|}$ of an instance i. Subsequently, for every label a meta level binary classifier is used to classify, resulting in a prediction of $\lambda_{1\dots|D|}$. To reduce the noise, ϕ correlations between all labels in the original data is computed. For the meta level, only labels exceeding the correlation threshold are used. By this, the dimensionality of the feature space in the meta-level and also the noise will be reduced.

Classifier Chains Classifier Chains by Read *et al.* [26] handles the label correlations by using a chain of single-label classifiers. Each of them is predicting the labels subsequently, e.g. for every label λ in the label set L a single-label classifier $C_{1\dots|D|}$ is trained. Every member in the chain adds feature with the confidence of the prediction for the current label to the feature space. Thus every classifier can use the prediction of the previous labels. The order of the chain influences the performance, nevertheless current implementations are using a random order. The performance can be improved by using ensembles of randomly ordered Classifier Chains.

Algorithm-adoption methods

ML-kNN ML-kNN proposed by Min-Ling Zhang and Zhi-Hua Zhou [34] is an adaption of the k-Nearest-Neighbor algorithm. As first step the k nearest neighbors are calculated, subsequently the label sets are mapped using the maximum-a-posteriori principle (MAP) based on the information gain about

the label sets of the neighboring instances. The method is described in more detail in algorithm 1.1. Given a dataset $D' = \{(y_i, Y_i), i = 1 \ldots |D|)\}$ and label set $L = \lambda_{1 \ldots |D|}$, a *membership counting* vector $\vec{C}_I(l)$ is used representing the number of neighbors of instance I which are associated with label λ. $N(I)$ is the set of the k nearest neighbors of instance I. H_1^λ is the event that t is associated with label λ in contrast H_0^λ is the event that I is not associated with label λ. With $E_j^\lambda : j = 0..k$ the event that among the k nearest neighbors are exactly j instances labeled with label λ. With those variables for each instance i a *category vector* $\vec{y}_I(\lambda)$ can be computed where $\vec{y}_I(\lambda)$ is 1 if λ is among the predicted label set $Y_{I_{pred}}$ for instance I. The vector $\vec{r}_I(\lambda)$ is holding numeric values to generate a ordering of the labels in order to predict a ranking instead a fixed label set.

Algorithm 1.1 ML-kNN

Require: $I = (x_i, Y_i) \in D$ {input is an instance I}
  Identify $N(I)$ {identify the k nearest neighbors}
  for $\lambda \in L$ **do** {compute for every label:}
    $\vec{C}_I(\lambda) = \sum_{a \in N(I)} \vec{y}_{x_a}(\lambda)$ {membership vector}
    $\vec{y}_I(\lambda) = \arg\max_{b \in \{0,1\}} P(H_b^\lambda) P(E^\lambda_{\vec{C}_I(\lambda)} | H_b^\lambda)$ {category vector}
    $\vec{r}_I(\lambda) = P(E^\lambda_{\vec{C}_I(\lambda)} | H_b^\lambda)$ {category vector with real values for ranking}
  end for
  return $\vec{y}_I$ and $\vec{r}_I$

HOMER

A multi-label learner introduced by Tsoumakas *et al.* [30] called HOMER is based upon a Hierarchy Of Multilabel classifERs. In order to build the hierarchy the label set L is split into k disjoint subsets. This is done recursively for each subset, until the number of remaining labels is $< k$. For every step in the hierarchy, clustering is used to find k clusters of similar labels. To ensure a balanced label distribution for each cluster, a modified cluster algorithm based on k-means, called *balanced k-means*, is used. For the clustering only the label part of data is used. The result is a tree-like hierarchy of L with $L_{root} = L$ and every single label as leaf. A meta-label level rules the path through the hierarchy. An instance I is annotated with meta-label μ_n if it is associated with any label of L_n. For every internal node, a classifier is trained, predicting current meta-label μ_n. Therefore, only branches with positive meta-labels will be considered for multi-label predicting, reducing the computational costs, because

for every branch a classifier has to deal with a much smaller label set. As classifiers have to deal with a much smaller set of labels in each level and only those branches with positive meta-labels are evaluated, a linear training and predicting complexity of $O(log_k(|L|))$ can be achieved.

2 Clustering and Feature Selection

Clustering and feature selection methods are core elements of this work. Thus, this chapter gives an introduction to the field and an overview over the used clustering and feature selection methods.

2.1 Cluster analysis

Cluster analysis or clustering is the task of assigning instances to subsets called clusters which are most similar inside and most dissimilar outside the cluster in respect to some distance or similarity measure. Clustering is an unsupervised method and does not use class information.

An application for clustering is gene expression analysis. Clustering methods are used to identify groups of genes with higher expression, than other groups. In line with the experiment, one can find families of genes which are correlated, and map co-expressions to diseases or biological pathways.

Many cluster methods require specifying the number of clusters prior to the process of clustering. This requires knowledge of the underlying structure of the dataset which is often not available or even the goal of the experiment. Cluster methods assign every instance to one particular cluster. However some cluster methods like Expectation Maximization (EM) compute membership probabilities. This is used to create a non-deterministic clustering of the data, allowing assigning cluster instances according to its membership probability.

There exists methods to evaluate a clustering. One of the most intuitive ones is to assign a class to each instance, separating the desired groups of instances. An optimal cluster method discriminates classes in different clusters. In this work, clustering is evaluated with respect to label dimensions, as density, cardinality and correlation as well to classification evaluation measures like accuracy.

Clustering will be used in order to find groups of correlated labels in multi-label datasets. Three types of clustering methods have been used.

2.1.1 Clustering Methods

Hierarchical Clustering

Hierarchical Clustering creates a hierarchy of clusters, either starting from one cluster and splitting, until every instance is its own cluster, called top-down or divisive clustering, or starting with every instance in a separate cluster and merging until all instances remain in one cluster, named bottom-up or agglomerative clustering [14].

Agglomerative clustering has a lower computational complexity as divisive clustering because splitting criteria can be computed directly, where most merging criteria need some "look-ahead" procedure.

Agglomerative clustering consists of three steps:

1. Compute a proximity matrix in-between each pair of clusters.
2. Merge the closest clusters.
3. Go to Step 1 until a minimum number of clusters are reached, or all instances remain in one cluster

Linkage criteria The linkage criteria describes how the distance between clusters is computed. In the following, the linkage criteria used in this work are presented.

1. Single linkage is the minimum distance between two clusters A and B
$$\min \{ d(a,b) : a \in A,\, b \in B \} \tag{2.1}$$
2. Complete linkage is the maximum distance between two clusters A and B
$$\max \{ d(a,b) : a \in A,\, b \in B \} \tag{2.2}$$
3. Average linkage is the mean distance of all elements in clusters A and B
$$\frac{1}{|A||B|} \sum_{a \in A, b \in B} d(a,b) \tag{2.3}$$
4. Average group linkage is the mean distance of all elements in the junction of clusters A and B (also called Mean Linkage)
$$\frac{1}{(|A|+|B|)(|A|+|B|-1)} \sum_{x,y \in A \cup B} d(x,y) \tag{2.4}$$

Measures For computing the proximity, any metric can be used. Fundamental ones are the Euclidean and Manhattan-Distance (also Taxicab-Distance):

$$D_{\text{Euclidean}}(x_i, x_j) := \sqrt{\sum_{k=1}^{p}(x_{ik} - x_{jk})^2} \tag{2.5}$$

$$D_{\text{Manhatten}}(x_i, x_j) := \sum_{k=1}^{p}|x_{ik} - x_{jk}| \tag{2.6}$$

Note: p is the length of the vector x.
The Chebyshev-Distance as the maximum distance of its elements:

$$D_{\text{Chebyshev}}(x_i, x_j) := \max_{k}^{p}(|x_{ik} - x_{jk}|) \tag{2.7}$$

Some feature selection methods produce a ranking of features. To apply hierarchical clustering on ranked features the Spearman Rank Correlation Coefficient [20] is used:

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \tag{2.8}$$

Note x_i,y_i are ranks, $\bar{x}$, $\bar{y}$ is the mean rank of x, respective y.
As there will occur no ties[1], in this application, a simpler formula introduced by Myers *et. al* can be used [20]:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2 - 1)} \tag{2.9}$$

where d_i is the difference between the ranks $d_i = x_i - y_i$ and n the number of examples.
Divisive clustering methods like DIANA [15] generally have higher computational complexity, thus they are not included in this work. Future work may explore more cluster methods as well as divisive clustering.

k-means Clustering

k-Means is another clustering algorithm based on distance metrics [18]. k-Means assigns examples to one of k clusters, as shown in algorithm 2.1.

[1] multiple elements sharing the same rank

Algorithm 2.1 k-means clustering

init $c_{1..k}$ as cluster centers randomly
while repeat $< n$ OR $c_{1..k}$ converge **do**
(2) assign each example x to the nearest cluster ($\min_i(d(c_i, x))$), where $d(x, y)$ is any metric
(3) recompute $c_{1..k}$ as the geometric centroid for each cluster
end while

After a randomly initialization of k cluster centers, each instance is assigned to its nearest cluster according to a distance measure. After assigning the instances the cluster center is recomputed. Step 2 & 3 is repeated at a most n times or until the center positions converge. To compute the distance $d(c_i, x)$ one can use any metric. It is faster than the Hierarchical Clustering as it does not need to update a full $n \times n$ distance matrix, but only the distance to the cluster center c_i. However, k-means converges to a local optimum. Solutions can still be improved by re-starts.

Expectation Maximization

The Expectation Maximization Algorithm has been introduced by Demp *et.al* [6]. It is an important tool for statistical analysis. Unlike hierarchical clustering and k-means clustering, EM is a probabilistic clustering algorithm. It models the data as a mixture of distributions. In an iterative process, parameters for those distributions are estimated, in case of a normal distribution those parameters are the expected value μ and variance σ^2. Each cluster is represented by one distribution. The distributions are initiated with randomly selected parameters. In alternating expectation and maximization steps, these parameters are optimized. The expectation step calculates the cluster probability for each instance, and the maximization step estimates the distribution parameters based on the cluster probabilities. Instances are assigned to the cluster with the highest membership probability. The clustering is finished when the log-likelihood of the instances belonging to the clusters saturates.

$$\begin{aligned} \log \prod_{x_i \in D} likelihood(x_i) &= \log \prod_{x_i \in D} \sum_{k \in Cluster} p_k P(x_i|C_k) \\ &= \sum_{k \in Cluster} \log \sum_{k \in Cluster} p_k P(x_i|C_k) \end{aligned} \tag{2.10}$$

The EM algorithm has the advantage that it does not depend on distance metrics or similarity measures. It also can handle missing values, which

would either be ignored or lead to high distances in other methods. On the other hand, its convergence often is suboptimal because it can stuck local optimums. Implementations overcome this problem by re-sampling, needing many EM-steps and increasing computation time.

2.2 Feature Selection

Datasets are constantly growing in both dimensions: attributes and instances. Thus classification must deal with a high number of possible redundant or irrelevant features [35]. Those interfering features heavily impact the classifying performance, as they tend to produce over-fitted models. Also, the growing dimensionality of data highly affects the running time of data-mining methods. Feature selection methods try to deal with those features by reducing the feature space and thus optimizing the learning process.

Unsupervised Feature Selection methods do not use information about the class of the instance. Supervised feature selection methods do take class information into account. Three classes of supervised feature selection methods can be distinguished: Wrappers, Embedded and Filter approaches. Wrappers use a learning scheme to evaluate features, by learning a classifier on a feature set and evaluating the prediction performance. They have high risk of over-fitting the model to the dataset and classifier method. They are also comparatively slow as they need to train a classifier for every evaluation step. Embedded feature selection methods are integrated into a classifying algorithm itself, e.g. decision tree where the nodes are built from the most discriminative features. Filters compute some score for each feature with respect to the class using different measures like e.g. information gain.

Additionally, filter methods can be divided by their output: Ranking methods and subset evaluators. Ranking methods aim to rank features according to a score and removing those which to not exceed a certain threshold. The second family finds groups of features for the optimal subset but does not evaluate the score of individual features.

Common to all methods is the goal to find features, which produce a good discrimination between classes.

2.2.1 Feature Selection Methods

Information Gain

Information Gain (IG) [4] is an intuitive and computationally simple evaluation measure. It calculates how much additional information is gained by using a feature X in respect to a label Y:

$$IG(X,Y) = H(X) - H(X|Y) \tag{2.11}$$

H is the entropy, which is a measure of uncertainty within a random variable.

$$H(X) = -\sum_i P(x_i)\log_2(P(x_i)) \tag{2.12}$$

$$H(X|Y) = -\sum_j P(x_j)\sum_i P(x_i|y_i)\log_2(P(x_i|y_i)) \tag{2.13}$$

Information Gain is the value by which the uncertainty of a label Y is in-/decreased when an additional feature X is added. A higher value indicates a higher relevance of the feature X. Information Gain is a univariate evaluation measure i.e. it does not take combinations of features into account.

Relief

Relief is an univariate filtering approach introduced by Kira and Rendell [16]. Its score is defined as shown in equation 2.14. A distance measure $d(x)$ is used to compute the distance of a instance I in a set of randomly selected p instances, and its nearest neighbors with same class f_{NH} (near hit) and different class f_{NM} (near miss) according to the feature value f_i.

$$SC_R(f_i) = \frac{1}{2}\sum_{t=1}^{p} d(f_{t,i} - f_{NM(x_t),i}) - d(f_{t,i} - f_{NH(x_t),i}) \tag{2.14}$$

Thus the feature is weighted by its ability to distinguish between classes. The score will increase if the distance to the closest neighbor with the same class is small and the distance to the closest neighbor with a different class is large.

Correlation based Feature Selection

The Correlation based Feature Selection (CFS) method is based on the idea that good feature subsets contain features which are highly correlated with

the label, and yield low correlation to each other. In order to find the best feature subset a merit score for each subset is computed:

$$Merit_S = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{2.15}$$

k is the number of features, the mean correlation of the feature set is:

$$\overline{r_{cf}} = \sum_{f_i \in S} \frac{1}{k} \sum cor(f_i, c) \tag{2.16}$$

and $\overline{r_{ff}}$ is the average correlation of features:

$$\overline{r_{ff}} = \sum_{f_i \in S} \frac{1}{k} \sum_{f_j \in S} cor(f_i, f_j) \tag{2.17}$$

Note that the correlation $cor(f_i, c)$ and respective $cor(f_i, f_j)$ use a normalized Information Gain measure called Symmetrical Uncertainty.

Other Feature Selection Methods

This work uses two more methods were for evaluation of different feature selection methods which are described briefly in the following:

Symmetric Uncertainty Symmetric Uncertainty is based on the uncertainty coefficient (equation 2.18) [22] which is closely related to information gain. C_{XY} and C_{YX} must not be necessarily equal, therefore, symmetrically uncertainty (equation 2.19) [32] computes the weighted average of the two uncertainty coefficients :

$$C_{XY} = \frac{IG(X;Y)}{H(Y)} \quad \text{and} \quad C_{YX} = \frac{IG(X;Y)}{H(X)} \tag{2.18}$$

$$U(X,Y) = 2\frac{I(X;Y)}{H(X) + H(Y)} \tag{2.19}$$

Significance Attribute Evaluation Significance Attribute Evaluation [1] is based on the idea that there is a strong possibility that instances with complementary values will belong to complementary classes. The significance is computed as a two-way function of the association of an attribute to the class decision.

3 Feature and Label Selection for Multi-Label Classification

Purpose of this work is to develop and evaluate methods for splitting multi-label datasets into subsets. Two general approaches have been developed:

1. Cluster based Splitting: using a transposed dataset for clustering, clusters resolve directly into groups

2. Feature- and Cluster based Splitting: clustering is performed on feature selection results and merged into groups.

A group is a dataset containing only a subset of attributes which can be features or labels from the original dataset. It can be seen as a vertical cutting of the dataset. In the following both methods will be explained in more detail.

3.1 Cluster based splitting

CML transposes the dataset D into a dataset D^T where every attribute column $A = X \cup L$ becomes an instance of D^T. In the next step, a cluster algorithm is applied to the dataset D^T, resulting in clusters containing attributes of D.

Thus label values are binary, in contrast to the features which can be numeric values, a preprocessing step normalizes all values.

In the clusters, similar features and labels are combined. With respect to the clustering method, the similarity measure will be a distance metric used by hierarchical clustering and k-means, or a probability distribution as used by EM. Attributes in the same cluster have a higher dependence between the cluster members than to the rest of the dataset.

Attributes can be split into features and labels, forming subsets $D_{1\ldots|N|} \subset D$ of the entire dataset D, where N is the number of clusters. On each subset $D_{1\ldots|N|}$ a multi-label learner is trained and predictions for the subsets of labels are taken together to get a prediction of the entire label set.

The procedure of training and predicting is shown in algorithm 3.1 and 3.2.

Algorithm 3.1 CML: training on dataset D

Require: dataset $D = \{(x_i, Y_i)|i = 1...|D|\}$
$D^T \leftarrow transpose(D)$ {transposing the dataset}
$C \leftarrow cluster(D^T)$ {clustering the data, C is the set of clusters}
$D^T_{1...N} \leftarrow split(D^T, C)$ {creating subsets of the data by splitting the feature and label space according to the clusters C}
$F_{1...N} \leftarrow train(D^T_{1...N})$ {training N multi-label learner $F_{1...N}$ on the subsets}

Algorithm 3.2 CML: predicting on dataset D

Require: I as unseen instance from D
Require: C as clusters from training
Require: $F_{1...N}$ trained multi-label learner for data subsets
$I_{1...|N|} \leftarrow split(I, C)$ {split instance according to the clusters of attributes}
for $k = 1 \rightarrow |N|$ **do**
$\vec{v_k} \leftarrow F_k(I_k)$ {prediction of particular label subsets}
end for
return $\vec{v} \leftarrow compose(v_{1...|N|})$ {composing vectors $v_{1...|N|}$ into a single vector}

3.2 Feature- and Cluster based Splitting

As first method, CML is based on the raw attribute values and it cannot qualify attributes with respect to a label. CML groups attributes which are sharing common values among the instances, but does not take into account their relevance to each other.

A second method, called FCML, was developed, in order to overcome this shortcoming. FCML takes into account the relevance of attributes in respect to a label by availing feature selection methods.

For every label $\lambda \in L$ of the dataset $D = \{(x_i, Y_i)|i = 1...|D|\}$, a feature selection is performed as described in section 2.2. The result is a matrix $D_{FS} = (x_{ij})_{i=1..l,j=1..m}$ where every row i represents a label, every column j represents an attribute, which can be a feature or a label, and x_{ij} is the score computed by the feature selection.

Subsequently, a cluster method is applied. Note that instances in D_{FS} represent labels of D, so every cluster c is a partition of L. Features are added if their score within any instance $I \in c$ exceeds a threshold.

This procedure results in sets of disjoint labels but may contain overlapping feature-spaces. Splitting the dataset along those attribute sets creates groups which can be trained by a multi-label learner.

Labels are clustered based on feature scores, which can be seen as weight of the features impact on the decision about the label. Thus, labels in the same cluster share the same designating features. Features are only added if they exceed a threshold, to avoid adding unnecessary or disturbing features, with more restrictive thresholds to concentrate on few high influencing features.

The procedure of feature selection and clustering is showed in algorithm 3.3 and 3.4.

Algorithm 3.3 FCML: FeatureSelection

Require: dataset $D = \{(x_i, Y_i)|i = 1...|D|\}$
$l \leftarrow |X \cup L|$ as the number of attributes
$m \leftarrow |D|$ as number of instances
$D_{FS} \leftarrow \mathbb{R}^{l \times m}$ as empty $l \times m$ matrix
for all $\lambda_i \in L$ **do** {feature selection for all labels}
$d_{FS_i} \leftarrow (score_{\lambda_i X_1}, ..., score_{\lambda_i X_l}) \leftarrow featureSelection(\lambda_i)$ {feature selection for λ_i outputs a vector with scores for every attribute $X_{1...l}$}
end for
return D_{FS}

Algorithm 3.4 FCML: Clustering

Require: D_{FS} matrix with scores for each label/attribute combination
$C \leftarrow cluster(D_{FS})$ {clustering on the feature selection score matrix}
Require: $G \leftarrow C$ as set of groups, initialized with the labels taken together by clusters
for all $c \in C$ **do** {clusters}
for all $\lambda_i \in c$ **do** {instances in cluster c (labels)}
for all $X_j \in d_{FS_i}$ **do** {attributes in score vector (attributes)}
if $d_{FS_{ij}} > threshold \wedge X_j \notin L$ **then** {only if X_j is not a label}
add X_j to c
end if
end for
end for
end for
return G groups containing subsets of labels and features

4 Experiments

This chapter presents implementation details of this work. Also datasets and evaluation measures used in this work are shown. The main part of this chapter presents the results of the clustering and feature selection components as well as results for the complete method.

4.1 Implementation

All methods are implemented in Sun Java Version 6[1]. As basic library MULAN[2] is used [31]. MULAN is a java library for multi-label learning based on the WEKA[3] machine learning software [12].

In figure 4.1 a class diagram of the implementation is shown. *MultiLabelLearnerBase* is the base class for all multi-label learners within MULAN. For transformation-based learners, as described in section 1.3.1, an additional class called *TransformationBasedMultiLabelLearner* provides additional parameters as the specification of a single label learner. *TransformationBasedMultiLabelLearner* itself is a child of *MultiLabelLearnerBase*. Methods presented in this work are placed into the *TransformationBasedMultiLabelLearner* branch, as they transform the problem into smaller sub problems. The multi-label learner for both methods, CML and FCML, is called *GroupBasedMetaClassifier*. It refers to a learner called *FilteredMLLearner*, which acts as preprocessing step and filters the data by given feature and label subsets. *FilteredMLLearner* forwards only the group containing the specified feature and label attributes to the actual learner. MULAN specifications enforce datasets to contain at least 2 labels, but groups with only one label may be used. Hence, an additional WEKA single label classifier is needed by *FilteredMLLearner*.

The class *GroupBasedMetaClassifier* can use two classes for identifying groups: *CML* and *FCML*. The classes implement the clustering based and feature selection and clustering based methods as described in section 3.1 and 3.2. The classes *FMCL* and *CML* identify groups and return them as indexes

[1] http://www.java.com/en/
[2] http://mulan.sourceforge.net/
[3] http://www.cs.waikato.ac.nz/~ml/weka/index.html

of attributes. These indexes are passed by *GroupBasedMetaClassifier* to the *FilteredMLLearner* class, which splits the dataset and forwards the subsets to the learner.

By searching for labels not contained in a group, *GroupBasedMetaClassifier* also ensures that the entire label space is covered. If labels are found which are not member of a group, an additional group is created, containing the missed labels and all features. Also, groups which do not contain a label will be ignored. Current methods only provide groups with disjoint label sets. Anyway, *GroupBasedMetaClassifier* prediction method, can combine different votes for the same label. This is done by evaluating the "strongest" vote by the distance to an undetermined vote (confidence of 0.5). *GroupBasedLearner* also offers some methods for calculating statistics on datasets, like density or cardinality. All implementations provide full compatibility to MULAN.

4.2 Multi-Label Datasets

A dataset can be described as $N \times L \times M$ where N is the number of instances, L the number of labels and M the number of features. *Bibtex* and *Delicious* are very large datasets in both label set cardinality and number of instances. This leads to very long computational times for clustering and feature selection. Therefore, those datasets where only tested with a small number of settings.

In table 4.1 all datasets used in this work are presented. Additionally to their dimensions and type, the label cardinality ($LCard$) and density ($LDens$) [29] of the dataset are shown. Both values measure the distribution of labels associated with an instance.

$$LCard(D) = \frac{\sum_{i=1}^{N} |y_i|}{N} \tag{4.1}$$

Label Cardinality ($LCard$) (see equation 4.1) is the average number of labels associated with each instance.

$$LDens(D) = \frac{1}{N} LCard(\mathcal{D}) \tag{4.2}$$

Label Density ($LDens$) (see equation 4.2) is the average fraction of the label space associated with an instance.

In the following, datasets are described in more detail.

Yeast [8] is a small biological dataset mapping genes to subsets of 14 different biological functions.

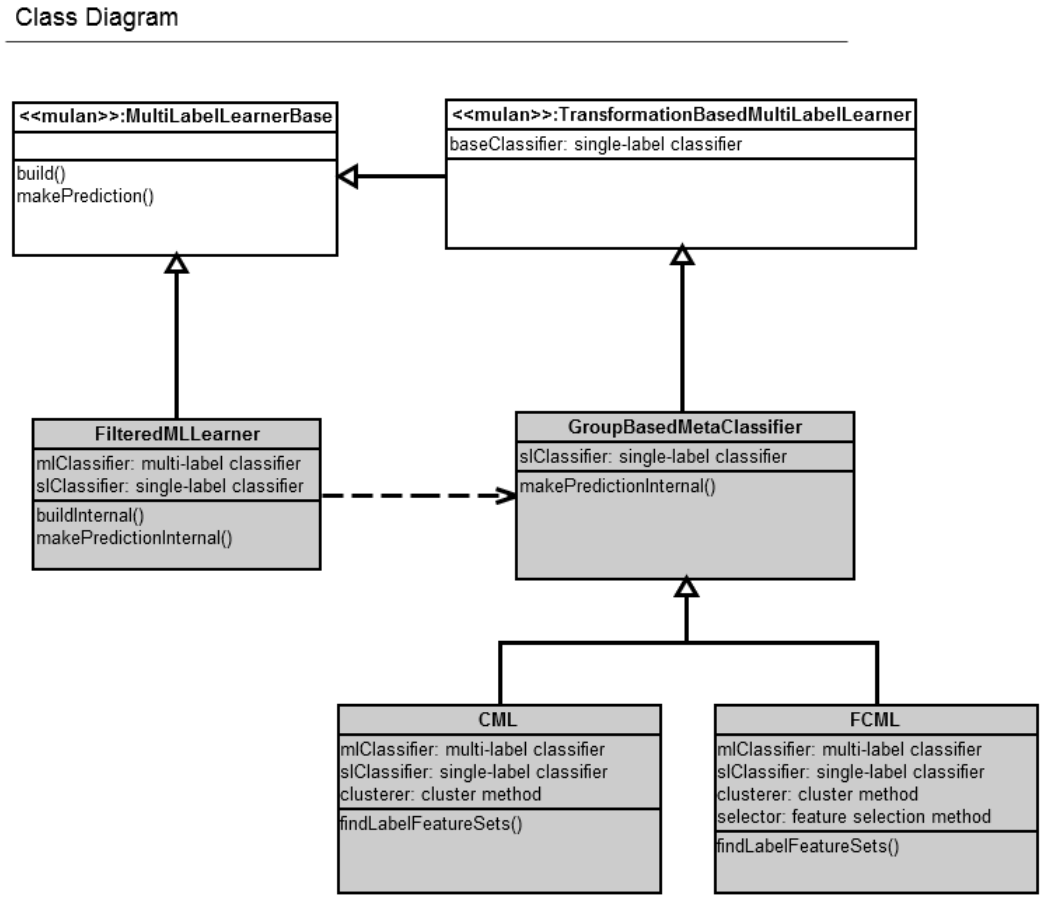


Figure 4.1: Class diagram showing the implementation structure of this work: self implemented classes are grey, white boxes represent MULAN classes. Most important input and output methods are shown. Drawn through lines indicate an ”extends”-relationship, dashed lines an ”used-by”-relationship. Main class is the *GroupBasedMetaClassifier* as it holds the subsets of the data ”groups” and the multi-label classifier as well the single-label classifier. For every group a *FilteredMLLearner* is used to extract the group subset from the dataset. For identifying groups either *CML* or *FCML* is used.

	N	L	M	$LCard$	$LDens$	type
Yeast	2417	14	$103n$	4.24	0.30	biol.
Medical	978	45	$1449b$	1.25	0.03	text
Enron	1702	53	$1001b$	3.38	0.06	text
Bibtex	7395	159	$1836b$	2.40	0.02	text
Delicious	16105	983	$500b$	19.02	0.02	text

Table 4.1: Multi-label datasets used in this work. N is the number of instances, L the number of Labels. M is the number of Features. n indicates numeric features, b binary features

Medical [21] was created for the Computational Medicine Centers 2007 Medical Natural Language Processing Challenge[4]. The instances are compiled out of brief medical reports containing symptom and their prognosis labeled with insurance codes.

Enron [24] is a subset of the the Enron e-mail corpus[5] hierarchically labeled with categories by the UC Berkley Enron Email Analysis Project[6].

Delicios [30] is a collection from the *del.ico.us* social bookmarking site[7]. Every posted site on this bookmarking site is annotated with different tags. Tsoumakas *et al.* filtered the set of overall 22139 tags to 983 frequent tags. In a second steps a boolean bag-of-word model for each bookmarked site was created and associated with the tags. In contrast to most other multi-label datasets in this context, labels were known prior to the feature space (bag-of-word model).

4.3 Evaluation Measures

Evaluation in multi-label classification differs from single-label evaluation: The latter is the simple decision if a prediction of a single class is right or wrong. The result can be easily stored in a confusion matrix. Measures like specificity, the fraction of true negatives in all negatives and sensitivity, the fraction of true positives in all positives had been known for a long time in this area.

[4] http://www.computationalmedicine.org/challenge/
[5] http://bailando.sims.berkeley.edu/enron/
[6] http://bailando.sims.berkeley.edu/enron_email.html
[7] http://del.ico.us

For multi-label problems, a strict right/wrong evaluation criteria would be very harsh as predictions of label sets would be counted as wrong if any false positive or false negative would occure, disregarding other correct predictions. Thus additional measures for multi-label prediction were introduced.

Two general fashions of multi-label evaluation measures exist: example-based measures where each predicted example is evaluated separately and label-based measures which are evaluating the binary relevance of each label individually [25].

A transfer of the classic single label *accuracy* into multi-label application inducts a measure $SubsetAccuracy(D)$ (equation 4.3) [10] as an example-based measure. $HammingLoss(D)$ (equation 4.4) [27] would be the label-based equivalent.

Example-based Evaluation Measures In this paragraph D is a dataset of testing, i.e. unseen examples. N denotes the number of examples, L the number of labels. $\hat{Y}_i$ is the predicted label set, Y_i is the known true label assignment.

$$SubsetAccuracy(D) = \frac{1}{N}\sum_{i=1}^{N} 1_{\hat{Y}_i = Y_i} \tag{4.3}$$

SubsetAccuarcy is the fraction of correct predicted label sets.

$$HammingLoss(D) = \frac{1}{NL}\sum_{i=1}^{N} |\hat{Y}_i \Delta Y_i| \tag{4.4}$$

HammingLoss is the failure (loss) in the predicted label sets averaged over instances N and labels L.

Other single label measures were adapted for multi-label use [11]:

$$Precision(D) = \frac{1}{N}\sum_{i=1}^{N} \frac{|\hat{Y}_i \vee Y_i|}{|\hat{Y}_i|} \tag{4.5}$$

Precision is the fraction of predicted labels which are relevant.

$$Recall(D) = \frac{1}{N}\sum_{i=1}^{N} \frac{|\hat{Y}_i \vee Y_i|}{|Y_i|} \tag{4.6}$$

Recall is the fraction of labels which are relevant and are also predicted.

[8] $A \Delta B$ is the symmetrical difference between A and B (logical XOR)
$0\Delta 0 = 1, 1\Delta 0 = 1, 0\Delta 1 = 1, 1\Delta 1 = 0$

$$F - Measure(D) = \frac{1}{N}\sum_{i=1}^{N}\frac{2|\hat{Y}_i \vee Y_i|}{|\hat{Y}_i| + |Y_i|} \tag{4.7}$$

F-Measure is the weighted average of precision and recall.

$$Accuracy(D) = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{Y}_i \wedge Y_i|}{|\hat{Y}_i \vee Y_i|} \tag{4.8}$$

Accuracy is the fraction of union and intersection of predicted and real label sets for each instance and averaged over the number of examples.

Label-based Evaluation Measures For label-based evaluation measures, any binary evaluation measure can be used. The only difference is the averaging process, as multi-label applications offer two ways of averaging: *macro-averaging* and *micro-averaging* [33]. A binary evaluation $M(tp, tn, fp, fn)$ uses the number of true positives (tp), true negatives (tn), false positives and false negatives (fp and fn) as input. Those numbers can be averaged as follows:

$$M_{macro} = \frac{1}{L}\sum_{i=1}^{L} M(tp_i, tn_i, fp_i, fn_i) \tag{4.9}$$

$$M_{micro} = \frac{1}{L} M(\sum_{i=1}^{L} tp_i, \sum_{i=1}^{L} tn_i, \sum_{i=1}^{L} fp_i, \sum_{i=1}^{L} fn_i) \tag{4.10}$$

4.4 Evaluation of Clustering- and Feature Selection Methods

In order to achieve best results for CML (section 3.1) or FCML (section 3.2), an ideal clustering and feature selection method has to be selected. Several algorithms and combinations were evaluated, each by a 5-fold cross-validation on various datasets. The numerous parameters for both, clustering and feature selection, lead to a very high number of evaluations with partially very long run times. Over 30.000 different settings were computed at the BIMComputeCluster[9] and the Linux-Cluster at the Leibniz-Rechenzentrum[10].

[9] http://www.bioinformatik-muenchen.de/studium/students/bim-compute-cluster

[10] http://www.lrz.de/services/compute/linux-cluster/

4.4.1 Clustering

Clustering is crucial for the performance of FCML and CML. This section shows different evaluation settings for clustering and its results.

For distance based cluster methods such as hierarchical clustering and simple k-means (section 2.1.1), a distance metric has to be chosen. Those methods also need the number of clusters nc to be specified. Furthermore, hierarchical clustering needs a linkage criterion to be given. This results in a very high number of evaluation settings. To average cluster performance, different feature selection methods and number of clusters, as well varying datasets are used.

CML In figure 4.2 clustering results for CML are shown. EM is outperformed by distance-based cluster methods. The EM algorithm is based on the Maximum-Likelihood-Principle, though for this application distance measures have advantages for the use as similarity measure for attributes. Regarding the distance measures, the *Chebyshev* distance shows disadvantages, while *Euclidean* and *Manhattan* distances seem to perform equally well. The latter ones are more sensitive to cumulative differences in many values, but less sensitive to single large differences. *Chebyshev* regards only the maximum distance between two instances, hence it is very receptive to outliers. Considering hierarchical cluster methods, *single, complete* and *average* linkage show equal performance. *Mean* has an average accuracy about 15% lower. Note that *Chebyshev* does not have disadvantages in *mean* linkage scenarios. *Mean*-linkage is also very sensitive to outliers, as the distance to every other instance is summed up.

FCML More detailed results for FCML are shown in figure 4.3. In comparison to CML, EM does not show a significant disadvantage versus distance based cluster methods. FCML does use scores computed from a feature selection method. Those scores are expectable to be distributed along some probability distribution. In the *medical* dataset EM shows improvements against hierarchical and k-means clustering. Despite this exception, no overall "best-performing" cluster method could be determined although hierarchical clustering shows enhancements in some cases. A closer look at the hierarchical clustering is done in figure 4.4 and 4.5. One can see again, there are no bold differences in neither, distance measure nor cluster method.

To overcome the problem of a missing standard setting working well for every scenario, a ranking for each scenario (dataset & multi-label learner) has

(a) ranking in dataset *CAL500*

rank	multi-label learner	feature-selection method	cluster algorithm	cluster method	distance measure
#1	ClassifierChain	InfoGainAttributeEval	SimpleKMeans		ManhattanDistance
#2	ClassifierChain	InfoGainAttributeEval	HierarchicalClusterer	SINGLE	ChebyshevDistance
#3	ClassifierChain	InfoGainAttributeEval	HierarchicalClusterer	SINGLE	EuclideanDistance

(b) ranking in dataset *enron*

rank method	multi-label learner distance measure	feature-selection method	cluster algorithm	cluster	
#1	ClassifierChain	InfoGainAttributeEval	HierarchicalClusterer	SINGLE	SpearmanCoefficient
#2	ClassifierChain	InfoGainAttributeEval	HierarchicalClusterer	MEAN	SpearmanCoefficient
#3	ClassifierChain	InfoGainAttributeEval	HierarchicalClusterer	COMPLETE	SpearmanCoefficient

(c) ranking in dataset *medical*

rank	multi-label learner	feature-selection method	cluster algorithm	cluster method	distance measure
#1	ClassifierChain	InfoGainAttributeEval	SimpleKMeans		EuclideanDistance
#2	ClassifierChain	InfoGainAttributeEval	SimpleKMeans		ManhattanDistance
#3	ClassifierChain	InfoGainAttributeEval	HierarchicalClusterer	SINGLE	ManhattanDistance

(d) ranking in dataset *yeast*

rank	multi-label learner	feature-selection method	cluster algorithm	cluster method	distance measure
#1	ClassifierChain	InfoGainAttributeEval	SimpleKMeans		EuclideanDistance
#2	ClassifierChain	InfoGainAttributeEval	SimpleKMeans		ManhattanDistance
#3	ClassifierChain	InfoGainAttributeEval	HierarchicalClusterer	SINGLE	ManhattanDistance

Table 4.2: Ranking of scenarios within different datasets using ClassifierChain as multi-label learner. Only the Top 3 results are shown. Different feature selection methods as well different cluster algorithm were used. The results are ordered along their example-based accuracy.

been created. Settings were ordered along their example-based accuracy. Tables 4.1(a), 4.1(b), 4.1(c) and 4.1(d) show the top 3 settings for the datasets *cal500*, *enron*, *medical* and *yeast*. Ranking differ between datasets but *HierarchicalClusterer* using single-linkage is among all top 3 results, thus further evaluation will use this setting.

Number of clusters Finally the parameter "number of clusters" has to be identified. The number of clusters (nc) can range from at least 2 to $|L|$, which results in each label to be clustered in a single cluster. Therefore, nc is highly related to the size of the dataset, where $|L|$ can be between 14 (*yeast*) and 983 (*delicious*).

For finding rules for determining nc, several attempts to identify correlations between nc and evaluation measures like *accuracy* or dataset specific measures like *label-density* were made. Figure 4.6 shows the result for the setting "ClassifierChain, Hierarchical Clustering, single-linkage, Euclidean Dis-

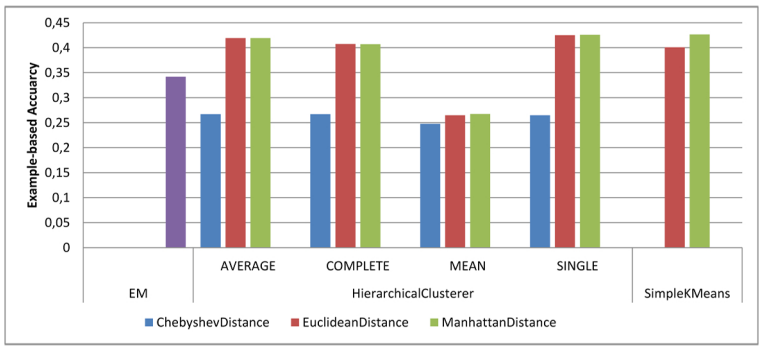
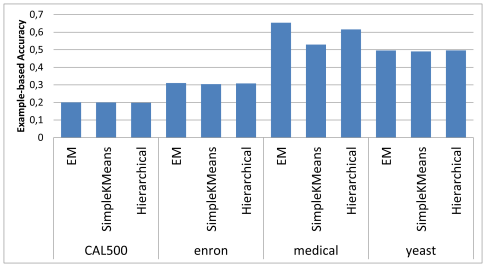
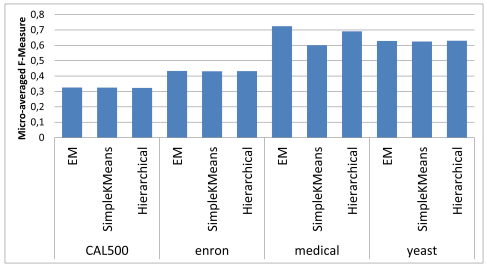


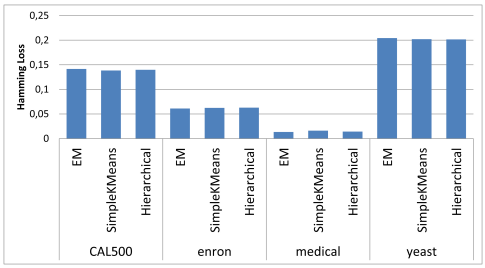
Figure 4.2: Average Example-based accuracy for different cluster algorithm on CML over different datasets



(a) Example-based Accuracy

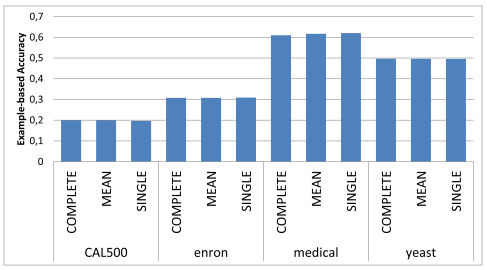


(b) Micro-averaged F-Measure

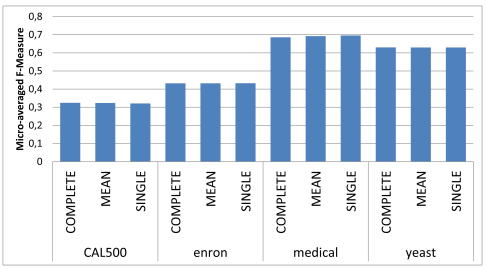


(c) Hamming Loss

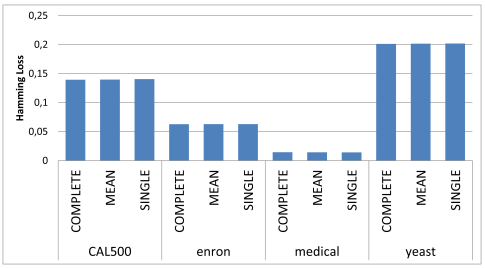
Figure 4.3: Evaluation of clustering algorithms in FCML. Different datasets were used. The measures are averaged over different settings, like nc, linkage criteria and distance measure



(a) Example-based Accuracy



(b) Micro-averaged F-Measure



(c) Hamming Loss

Figure 4.4: Evaluation of hierarchical clustering methods on different datasets. The measures are averaged over different distance measures

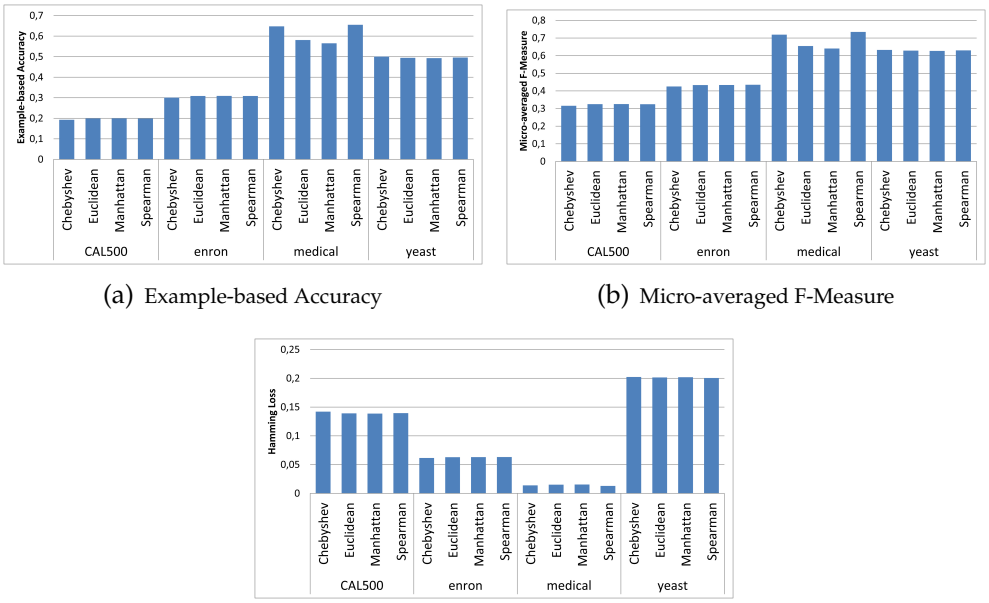
	density	labels per cluster	number of clusters
Accuracy	-0,2623	-0,195886517	-0,4528997
F-Measure	-0,214	-0,208032488	-0,47366346
HammingLoss	0,78642	0,063797033	0,23463825

Table 4.3: Pearson-correlation matrix for the setting "ClassifierChain, Hierarchical Clustering, single-linkage, Euclidean Distance, InformationGain" *CAL500* dataset with different nc

tance, InformationGain" on the dataset *CAL500*.

In table 4.3 a correlation matrix between the measures *Example-based Accuracy*, *Micro-averaged F-Measure*, *Hamming-Loss* and *avgDensity* as well the average labels per cluster and the number of clusters itself is shown. *avgDensity* is the density averaged over all groups. The correlation was computed from the values for each step for nc from 2 to $|L|$.

The matrix shows poor correlation, with an exception on "Hamming-Loss" to "avgDensity". As the best value in Hamming-Loss is 0, performance seems to decrease with higher average density. However, the reason for this relationship could not be covered in this work due to time limits.



(a) Example-based Accuracy

(b) Micro-averaged F-Measure

(c) Hamming Loss

Figure 4.5: Evaluation of distance measure using hierarchical clustering on different datasets. The measures are averaged over different distance measures

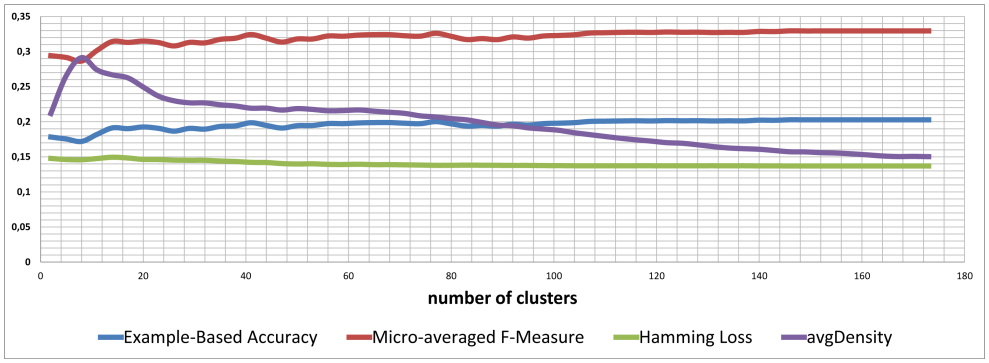


Figure 4.6: Example-based Accuracy, Micro-averaged F-Measure, Hamming Loss and average Density with growing number of clusters. "ClassifierChain, Hierarchical Clustering, single-linkage, Euclidean Distance, InformationGain" *CAL500* dataset with different nc.

4.4.2 Feature-Selection

FCML (see section 3.2) uses feature selection methods to compute scores/ranks for every attribute. The configuration of the feature selection scenario is crucial to the learning and prediction process.

In order to isolate effects of ranking, in detail to ignore scores within the adding features to groups, any threshold is set to minimum. This leads to groups which only differ in the label subsets but always contain the entire features space. The evaluation of an optimal feature selection threshold is a separate task.

To evaluate the feature selection process, a clustering algorithm is needed. To reduce the influence of good or bad clustering, various cluster algorithms and parameters were used. Finally, the performance of the FCML method is averaged.

To reduce the number of evaluations, two scenarios where considered: Firstly, five feature selection methods were evaluated on one dataset, namely *CAL500*. The results are shown in 4.7. Although *CfsSubsetEval* shows tiny improvements against other methods, no superior method could be determined.

A second step evaluates three different feature selection methods on varying datasets: *Relief*, *CFS* and *Information Gain*. Those three methods were chosen because they represent independent solution approaches. Results, as shown in figure 4.8, point out that CFS can outvalue both other methods only in the *medical* dataset. Hence, Hamming-Loss shows no improvement.

As no superior feature selection method could be found, further evaluation *information gain* uses because of its simplicity. It computes mutual certainty for a pair of attributes satisfying the claim that groups consist of attributes which have strong mutual associations.

Information gain has also advantages in terms of computing time. Therefore, it allows more settings to be evaluated. Nevertheless, further evaluating and parameter optimization could lead to better results on certain datasets.

4.4.3 Evaluation of methods

In the following, results from CML and FCML are presented. Note that CML was developed and tested prior to FCML, the latter one uses an extended set of datasets and evaluation techniques. Due to time limits it was not possible to repeat tests for CML with new datasets.

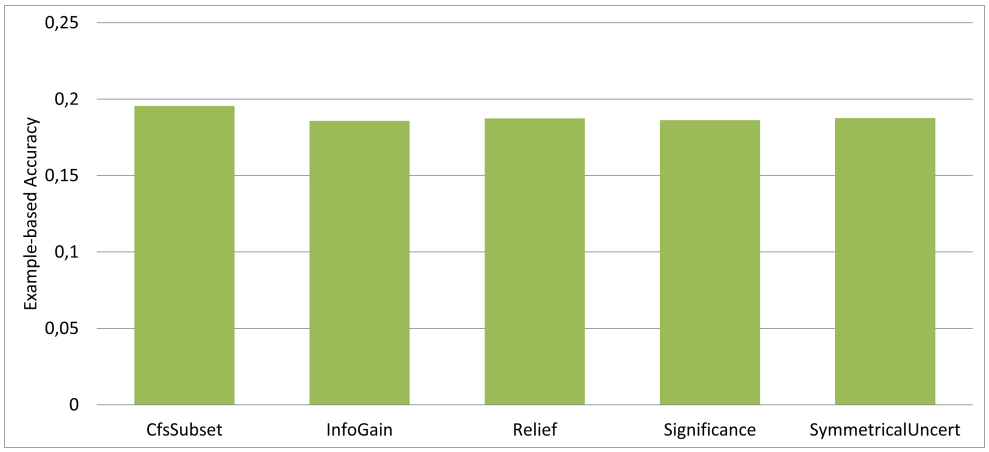
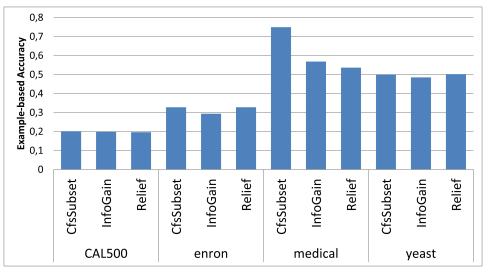
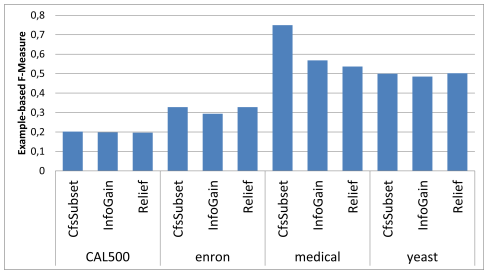


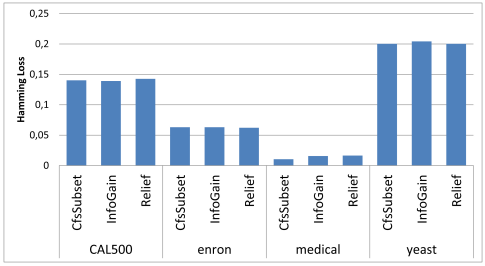
Figure 4.7: Feature-Selection Methods on the *CAL500* dataset. Example-based Accuracy is averaged over different clustering settings



(a) Example-based Accuracy



(b) Micro-averaged F-Measure



(c) Hamming Loss

Figure 4.8: *Relief*, *CFS* and *Information Gain* feature selection on different datasets

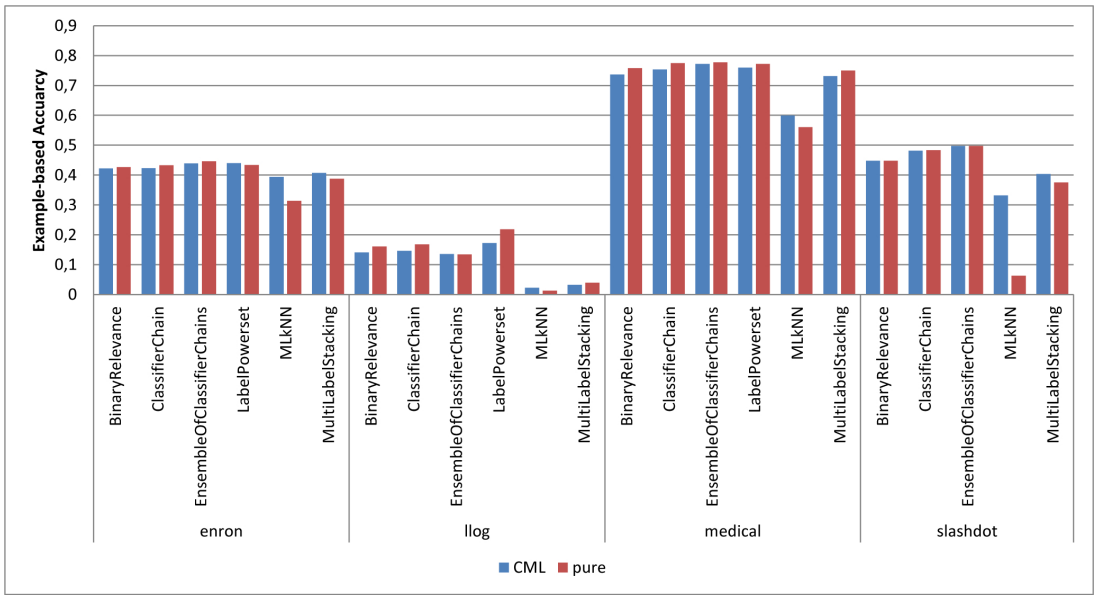


Figure 4.9: CML compared to pure multi-label learners using the example-based accuracy. For CML the best result is taken.

CML

In figure 4.8 results of CML compared to pure multi-label learners are shown. If needed by the multi-label learner, a Support-Vector Machine (SMO) was used. As described before a hierarchical clustering with Euclidean distance measure and single-linkage was used. For the number of clusters, nc, different number of clusters, namely $2, 4, 6$, where chosen. For the results the best nc was taken. All evaluation was done by a 5-fold cross-validation.

Although nearly every scenario shows similar results for CML and the pure learner, performance of ML-kNN could be improved in every case.

ML-kNN computes distances between instances and assigns query instance to the major label set of the k-nearest neighbors using a maximum-a-posteriori principle. Filtering the label space can enhance the MAP principle as the elements of the output label space are highly dependent and therefore MAP probabilities are likely to be high.

In summery, CML can enhance the performance of multi-label learners in some cases, but on average CML shows the same performance as its pure multi-label scheme.

FCML

Results for FCML are shown in figure 4.10. For every scenario (dataset + multi-label learner), all possible numbers of clusters nc were evaluated and the best result is presented. To reduce the complexity of parameters, feature selection was used without threshold, e.g. groups always contain the entire feature space. HOMER, as described in section 1.3.1 was used for comparison. HOMER uses a parameter k describing for the number of clusters at each hierarchical level. Every scenario, $k = 2 \dots 8$ was evaluated and again the best result is presented. However, HOMER did not finish in every scenario due to memory limits (6GB) or problems in the implementation[11]. In addition, for every scenario a pure multi-label learner has been trained.

FCML does not show superior results in Example-based Accuracy. Merely Hamming Loss could be reduced in some cases. Still, ML-kNN could be enhanced in some cases: *enron*, *medical*, *bibtex*. HOMER performs better in almost any case. Best results for FCML in this scenario are achieved with $nc = 158$ being a clustering of every label into a sole cluster which indicates there are very few dependencies in the label space.

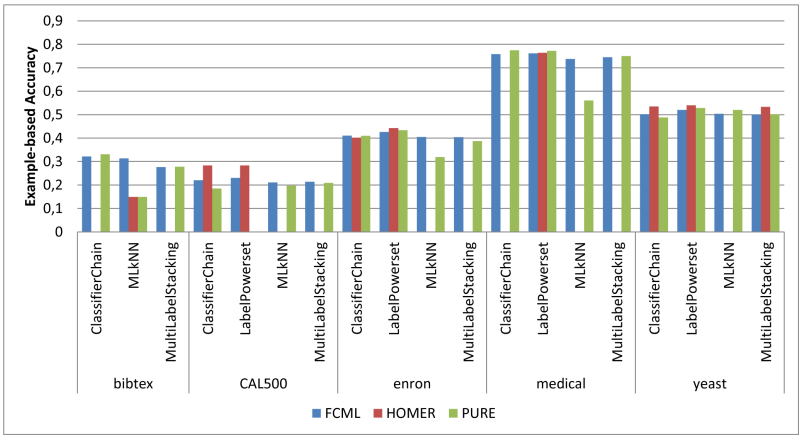
In terms of Hamming Loss or Micro-Averaged F-Measure, FCML can achieve small enhancements to the pure and HOMER multi-label learner. Both Hamming Loss and Micro-Averaged F-Measure take false positives into account, while accuracy does not. That indicates that FCML has slightly better performance in predicting absent labels compared to pure and HOMER. By reducing the label space and therefore reducing the influences of other labels, non-relevant and misleading signals are eliminated.

It is problem to identify the effect of nc on the performance as different multi-label learners perform best on very different nc: For example the *bibtext* dataset with $|L| = 158$, where Classifier Chains have the best result with a $nc = 8$, MlKNN $k = 158$ and MultiLabel-Stacking with $nc = 14$.

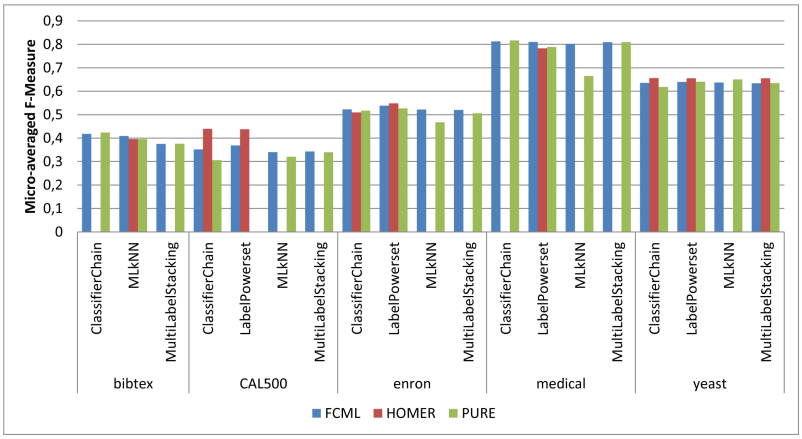
Summary

The results show that a clustering of labels into different groups and splitting the dataset according to those label sets does not cause a lower predicting performance. Even very strong splitting, where every label falls into a single cluster, may enhance the performance, what indicates that inter-label relations are not used. Prediction of absent labels could be slightly enhanced, indicated by lower Hamming Loss results. This is done by reducing the noise in the label-space by grouping them into smaller subsets.

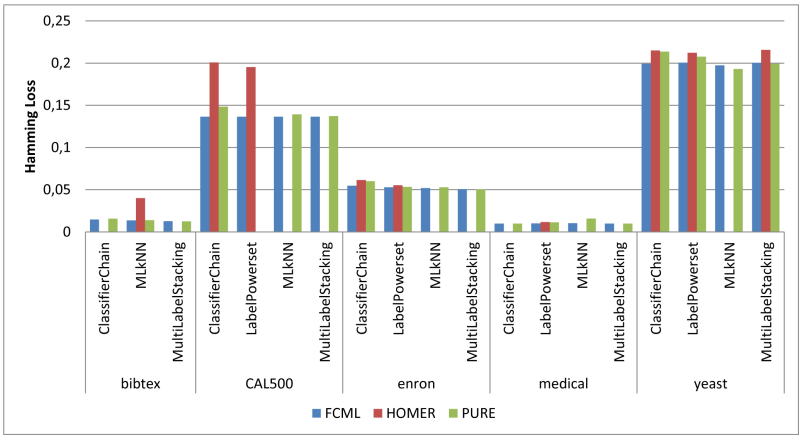
[11] IndexOutOfBounds-Exception during JAVA runtime



(a) Example-based Accuracy



(b) Micro-averaged F-Measure



(c) Hamming Loss

Figure 4.10: FCML compared to pure multi-label learners and HOMER on different datasets. In every setting the best result is presented. HOMER did not finish on all scenarios.

5 Conclusions

This work aims to find subgroups of labels and features in multi-label classification. It was expected to gain a significant enhancement of predicting performance. This claim could not be yet verified in its entirety.

However, results show that a splitting is possible without losing label information. Also small improvements in predicting true negative labels could be shown. Label dependencies in multi-label problems are still under research. Current work [5] shows that label dependencies may have to be considered in even more detail than done so far.

For identifying label groups, clustering methods are used. To enhance the clustering, feature selection algorithms have been applied, increasing the focus on attribute relations. For this, different cluster and feature-selection methods were evaluated. A framework for group-based multi-label classification has been implemented using the MULAN Java Library.

The idea of this work suffered from the complexity of its components, clustering and feature selection. Future work must aim to analyze the effects of the numerous parameters in more detail, than it could be done in the limited time of the bachelor thesis. Finding the ideal number of clusters is still subject of research in the field of unsupervised clustering. For example, the datasets could be evaluated in terms of density, label distribution and label correlation. Also, it would be a beneficial to find a rule to specify the number of clusters nc dependent to the label space. In addition, the feature selection process could be evaluated in more detail.

Also, it seems to be promising to develop some horizontal splitting, where groups are not only subsets of attributes but also subsets of instances with high specificity to certain label (sub)sets. New bi-clustering algorithms seem to be a suitable solution for this task.

In conclusion, first steps towards for Label- and Feature-Selection in Multi-Label Classification were taken. Due to time limitations not all aspects of involved methods could yet be treated in full detail, which may achieve improved performance. However, this work may serve as a starting point to future work in Label and Feature Selection in Multi-Label Classification.

Bibliography

[1] A. AHMAD and L. DEY, *A feature selection technique for classificatory analysis*, PhD thesis, Indian Institute of Technology, October 2004.

[2] A. BAIROCH and R. APWEILER, *The SWISS-PROT protein sequence data bank and its new supplement TREMBL*, Nucleic Acids Res, 24(1), 1996, pp. 17–21.

[3] C. CORTES and V. VAPNIK, *Support-vector networks*, Machine Learning, 20(3), 1995, pp. 273–297.

[4] T. M. COVER and J. A. THOMAS, *Elements of Information Theory*, John Wiley & Sons. Inc., Hoboken, NJ, US, 1991.

[5] K. DEMBCZYŃSKI, W. WAEGEMAN, W. CHENG, and H. E., *On label dependence in multi-label classification*, in Workshop Proceedings of Learning from Multi-Label Data, 2010, pp. 5–12.

[6] A. P. DEMPSTER, N. M. LAIRD, and D. B. RUBIN, *Maximum Likelihood from Incomplete Data via the EM Algorithm*, Journal of the Royal Statistical Society Series B (Methodological), 39(1), 1977, pp. 1–38.

[7] D. J. DIX, K. A. HOUCK, M. T. MARTIN, A. M. RICHARD, R. W. SETZER, and R. J. KAVLOCK, *The ToxCast program for prioritizing toxicity testing of environmental chemicals.*, Toxicological Scienes, 95(1), 2007, pp. 5–12.

[8] A. ELISSEEFF and J. WESTON, *A kernel method for multi-labeled classification*, in Advances in Neural Information Processing Systems, 2001, pp. 681–687.

[9] U. FAYYAD, G. PIATETSKY-SHAPIRO, and P. SMYTH, *From data mining to knowledge discovery in databases*, AI Magazine, 17, 1996, pp. 37–54.

[10] N. GHAMRAWI and A. MCCALLUM, *Collective multilabel classification*, in Proceedings of the 14th ACM international conference on Information and knowledge management, 2005, pp. 195–200.

[11] S. GODBOLE and S. SARAWAGI, *Discriminative Methods for Multi-Labeled Classification*, in Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2004, pp. 22–30.

[12] M. HALL, E. FRANK, G. HOLMES, B. PFAHRINGER, R. REUTEMANN, and I. H. WITTEN, *The WEKA data mining software: an update*, SIGKDD Explorations, 11(1), 2009, pp. 10–18.

[13] INTERNATIONAL HUMAN GENOME SEQUENCING CONSORTIUM, *Initial sequencing and analysis of the human genome*, Nature, 409(6822), 2001, pp. 860–921.

[14] A. K. JAIN, M. N. MURTY, and P. J. FLYNN, *Data Clustering: A Review*, ACM Computing Surveys, 31, 1999, pp. 264–323.

[15] L. KAUFMAN and P. J. ROUSSEEUW, *Finding Groups in Data An Introduction to Cluster Analysis*, Wiley Interscience, New York, NY, USA, 1990.

[16] K. KIRA and L. A. RENDELL, *A Practical Approach to Feature Selection*, in Ninth International Workshop on Machine Learning, 1992, pp. 249–256.

[17] N. M. LUSCOMBE, D. GREENBAUM, and M. GERSTEIN, *What is Bioinformatics? A Proposed Definition and Overview of the Field*, Methods of Information in Medicine, 40(4), 2001, pp. 346–358.

[18] J. B. MACQUEEN, *Some Methods for Classification and Analysis of MultiVariate Observations*, in Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.

[19] T. M. MITCHELL, *Machine Learning*, McGraw-Hill, Inc., New York, NY, USA, 1st ed., 1997.

[20] J. L. MYERS and A. D. WELL, *Research Design and Statistical Analysis*, Lawrence Erlbaum Associates, London, UK, 2ncs ed., 2003.

[21] J. PESTIAN, C. BREW, P. MATYKIEWICZ, D. J. HOVERMALE, K. B. COHEN, and D. WLODZUSLAW, *A shared task involving multi-label classification of clinical free text*, in BioNLP '07 Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing, 2007, pp. 97–104.

[22] W. H. PRESS and F. B. P., *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, New York, NY, USA, 2nd ed., 1988.

[23] J. R. QUINLAN, *Induction of Decision Trees*, Machine Learning, 1, 1986, pp. 81–106.

[24] J. READ, *A Pruned Problem Transformation Method for Multi-label classification*, in Proceedings of the New Zealand Computer Science Research Student Conference, 2008, pp. 143–150.

[25] J. READ, *Scalable Multi-label Classification*, PhD thesis, University of Waikato, September 2010.

[26] J. READ, B. PFAHRINGER, G. HOLMES, and E. FRANK, *Classifier Chains for Multi-label Classification*, in Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II, 2009, pp. 254–269.

[27] R. E. SCHAPIRE and Y. SINGER, *BoosTexter: A Boosting-based System for Text Categorization*, Machine Learning, 39, 2000, pp. 135–168.

[28] G. TSOUMAKAS, A. DIMOU, E. SPYROMITROS-XIOUFIS, V. MEZARIS, I. KOMPATSIARIS, and I. VLAHAVAS, *Correlation-Based Pruning of Stacked Binary Relevance Models for Multi-Label Learning*, in Proceedings of the 1st International Workshop on Learning from Multi-Label Data, 2009, pp. 101–116.

[29] G. TSOUMAKAS and I. KATAKIS, *Multi Label Classification: An Overview*, International Journal of Data Warehousing and Mining, 3(3), 2007, pp. 1–13.

[30] G. TSOUMAKAS, I. KATAKIS, and I. VLAHAVAS, *Effective and Efficient Multilabel Classification in Domains with Large Number of Labels*, in Proceedings ECML/PKDD 2008 Workshop on Mining Multidimensional Data, 2008.

[31] G. TSOUMAKAS, E. SPYROMITROS-XIOUFIS, J. VILCEK, and I. VLAHAVAS, *Mulan: A Java Library for Multi-Label Learning*, Journal of Machine Learning Research, 12, 2011, pp. 2411–2414.

[32] I. H. WITTEN, E. FRANK, and M. A. HALL, *Data mining : Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd ed., 2011.

[33] Y. YANG, *An evaluation of statistical approaches to text categorization*, Journal of Information Retrieval, 1(1/2), 1999, pp. 67–88.

[34] M. ZHANG and Z. ZHOU, *A k-nearest neighbor based algorithm for multi-label classification*, in IEEE International Conference on Granular Computing, 2005, pp. 718–721.

[35] Z. ZHAO, F. MORSTATTER, S. SHARMA, S. ALELYANI, A. ANAND, and H. LIU, *Advancing Feature Selection Research - ASU Feature Selection Repository*, tech. rep., School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, 2010.