12-2018

Multi-Label Classification Using Higher-Order Label Clusters

Dilanga Lakshitha Bandara Abeyrathna Galapita Mudiyanselage
University of Nebraska at Omaha

Multi-Label Classification Using Higher-Order Label Clusters

A Thesis

Presented to the

Department of Computer Science,

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfillment
of the Requirements for the Degree

Master of Science in Computer Science

University of Nebraska at Omaha

by

Dilanga Lakshitha Bandara Abeyrathna, Galapita Mudiyanselage

December, 2018

Supervisory Committee:

Dr. Parvathi Chundi

Dr. Mahadevan Subramaniam

Dr. Abhishek Parakh

Multi-Label Classification Using Higher-Order Label Clusters

Dilanga Galapita Mudiyanselage, M.S.

University of Nebraska, 2018

Adviser: Dr. Parvathi Chundi

Multi-label classification (MLC) is one of the major classification approaches in the context of data mining where each instance in the dataset is annotated with a set of labels. The nature of multiple labels associated with one instance often demands higher computational power compared to conventional single-label classification tasks. A multi-label classification is often simplified by decomposing the task into single-label classification which ignores correlations among labels. Incorporating label correlations into classification task can be hard since correlations may be missing, or may exist among a pair or a large subset of labels. In this study, a novel MLC approach is introduced called *Multi-Label Classification with Label Clusters (MLC–LC)*, which incorporates label correlations into a multi-label learning task using label clusters. *MLC–LC* uses the well-known *Cover-coefficient based Clustering Methodology* (C^3M) to partition the set of labels into clusters and then employs either the *binary relevance* or the *label powerset* method to learn a classifier for each label cluster independently. A test instance is given to each of the classifiers and the label predictions are unioned to obtain a multi-label assignment. The C^3M method is especially suited for constructing label clusters since the number of clusters appropriate for a label set as well the initial cluster seeds are automatically computed from the data set. The predictive of *MLC–LC* is compared with many of the matured and well known multi-label classification techniques on a wide variety of data sets. In all experimental settings, *MLC–LC* outperformed the other algorithms.

ACKNOWLEDGMENTS

Firstly, I am very grateful for my advisor Dr. Parvathi Chundi, for all support and help she bestowed on me. Dr. Chundi extended her support not only to improve my research but funded my master's studies as a Graduate Research Assistant. This assistantship gave me a scope to extend my research work and gave me an opportunity to work in multiple projects which helped me gain enormous knowledge and opened me to a wide variety of new areas in technology. I express my deep gratitude for Dr. Chundi for the educational, financial and moral support from the beginning of my degree.

I would like to acknowledge Dr. Mahadevan Subramaniam, member of the supervisory committee who I worked closely with, has always been a pillar of strength all through my master's journey. I am grateful that I got an opportunity to work with Dr. Subramaniam, who has always been enthusiastically motivating me with not only my research but also guided me through tough times in my personal life and has been a great moral support. I would like to thank Dr. Abhishek Parakh, member of my supervisory committee for his continuous support for my research and actively involving in improving my research.

I am thankful for my mate Ravi Teja, who helped me with the implementation of C^3 clustering in Java that was greatly helpful in my research. I would like to thank my fellow students Srikanth Vadla, Vidya Bommanapally and Shane McDermott for helping me in several ways and providing me with the gaming platform *Quasim* where I could readily apply for my research work.

My special thanks to Holland Computer Centre Resources and Crew, which is the main source of results in my research project, without which my thesis could have been very difficult.

This accomplishment would not have been possible without all your help. My sincere gratitude to all of you for having confidence in me and supporting me all through. Thank you!

Contents

List of Figures v

List of Tables vii

1: Introduction 1

1.1 Overview 1

1.2 Aims and Objectives 3

1.3 Structure of chapters 5

2: Literature Review 6

2.1 Learning Algorithms 6

2.2 Evaluation Matrices 9

2.2.1 Label-based evaluation matrices 10

2.2.2 Example-based evaluation matrices 11

2.3 Cover Coefficient Clustering method 12

3: METHODOLOGY 15

3.1 C^3M Clustering 15

3.2 MLC-LC 19

3.3 Computational Complexity of MLC–LC 21

4: EXPERIMENTAL SETUP 22

4.1 Data Sets 22

4.2 Results of Experiments 25

4.2.1 Classification Accuracy of *MLC–LC* 26

4.2.2 Classification Accuracy with Different Training Set Sizes 32

4.2.3 Classification with Different Label Clustering Algorithms 32

5: CONCLUSIONS AND FUTURE WORK **41**

References **43**

Appendices **46**

List of Figures

1.1 Beach scene 2

2.1 Symbols and Notations Used in this report 7
2.2 BR method 7
2.3 LP method 8
2.4 RAkEL method 8
2.5 Evaluation matrices categorization 10
2.6 D matrix representing document-term matrix 13

3.1 E matrix representing label assignment 16
3.2 The Cover Co-efficient Matrix 16
3.3 Non-overlapped label set training process 20
3.4 Classification process 21

4.1 micro-F_1, macro-F_1 and hamming Loss scores for 66% random sampling training set. Experimental results (mean$\pm$std) on data. $\uparrow$ ($\downarrow$) indicates the larger (smaller), the better 27
4.2 micro-F_1, macro-F_1 and hamming Loss scores for 66% stratified sampling training set. Experimental results (mean$\pm$std) on data. $\uparrow$ ($\downarrow$) indicates the larger (smaller), the better 28
4.3 Ranking of MLC methods according to their micro-F_1, macro-F_1 and hamming loss scores for 66% random sampling training set 29

4.4 Ranking of MLC algorithms Based on micro-F_1, macro-F_1 and hamming Loss score for stratified sampling 30
4.5 F_1 scores for Yeast data set with 66% random sampling training set and 34% testing set 31
4.6 micro-F_1, macro-F_1 Scores and hamming loss values of Different MLC methods for Training Set Sizes 70% and Testing set size 30%. 33
4.7 micro-F_1, macro-F_1 Scores and hamming loss values of Different MLC methods for Training Set Sizes 50% and Testing set size 50%. 34
4.8 micro-F_1, macro-F_1 scores and hamming loss values of Different MLC methods for Training Set Sizes 30% and Testing set size 70%. 35
4.9 (a) Micro-F_1 and (b) Macro-F_1 scores of the *MLC–LC* Classifiers Obtained from Different Clustering Methods 39

List of Tables

4.1 Multi-label Data Sets 23
4.2 Label Clustering using C3M, C3K, and C3H methods 38

Chapter 1

Introduction

1.1 Overview

According to the machine learning literature, there are three main classification categories namely *binary, multi-class,* and *multi-label* classification. This categorization is mainly based on the number of labels assigned to each instance in the dataset. When instances are associated with single label from a set which has two outcomes positive/negative (or true/false) can be recognized as binary classification where as in Multi- class classification problems each instance in the dataset annotated with single label from a finite set of outcomes. As an example, detecting a mail whether it is spam or not (true/ false) is a binary classification problem and categorizing Iris flower into either an Iris Setosa, Iris Versicolour, or Iris Virginica is a multi-class classification problem. However, these two classification problems are commonly recognized as *Single label* classification problem as final annotation is only having single label. Both of these classification problems learn a classifier (or a classifier ensemble) using a set of training instances and then use the classifier (or the ensemble) to assign a single label to a new (or a test) instance.

Multi-label classification (MLC) is an extension of the traditional single-label classification problem where each instance is associated with a set of labels. For example, according to the single label classification, Figure 1.1 can be annotated beach where as Multi-label classification approach can annotate the same image with beach, blue, water, chair, tree, sea for the image. Further, Multi-label classification problems are common in real-world applications. For example, a given email message may be labeled as both *personal* and *important*. Similarly, a news article may be classified as *religious, conservative,* and *liberal*. Each input instance in a multi-label data set may be labeled with more than one label. Hence, the challenge is to annotate newer instances with multiple appropriate labels as possible.

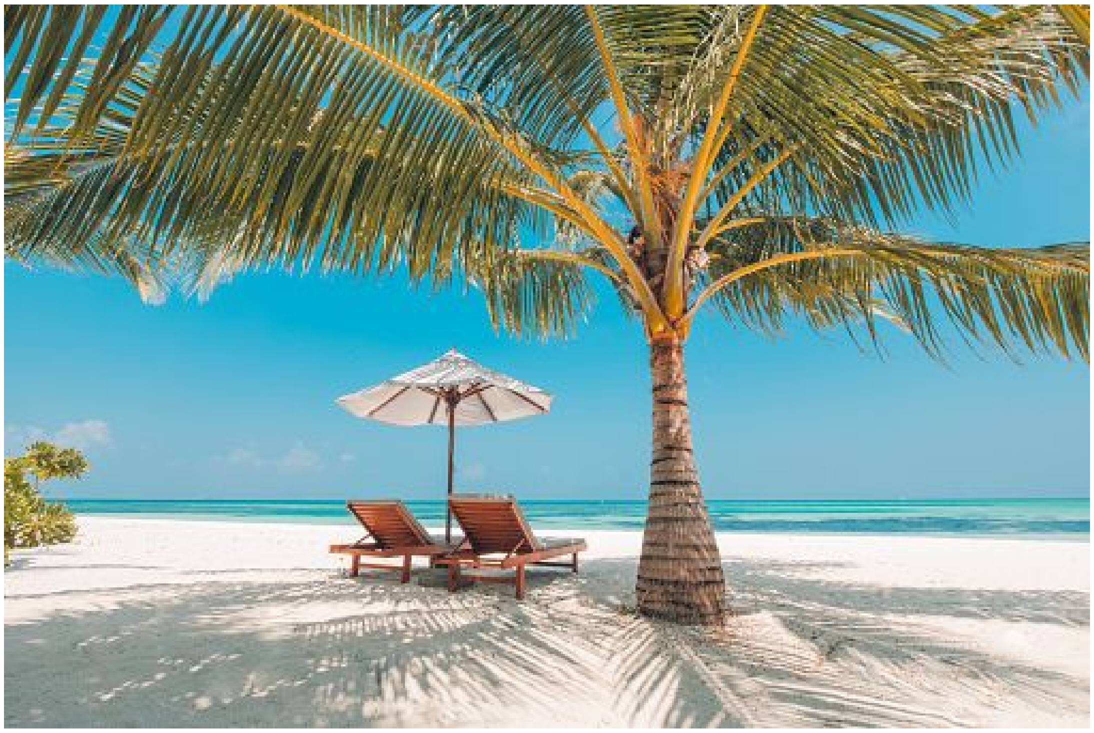


Figure 1.1: Beach scene

Mainly there are two approaches to tackle multi-label classification problem known as problem transformation method and algorithm adaptation method. Problem transformation method transfers a multi-label problem into a set of single label classification problems so that they can be handled by a set of single-label classifiers and union the outputs to retrieve the final solution. One simple problem transformation approach for predicting multiple labels for a given instance is to train a binary classifier for each label separately, which is also known as the *binary relevance (BR)* method[1] (Figure 2.2). A test instance is given to each binary classifier to find all appropriate labels.

However, using binary classification approach to solve a multi-label classification problem has severe drawback that it ignores the correlations among labels. For example, if an image (Ex: Figure 1.1) is labeled with *water*, it is more likely that it is labeled with *blue*. As these set of binary classifiers executes mutually exclusive manner there is no way to identify or preserve label dependencies. One way to incorporate correlations among labels is to treat each label combination as a distinct label and transform multi-label classification problem into multi-class classification problem, known as the *label powerset (LP)* method.[1] Here, each class label is a boolean vector of the size of all possible label set. If an instance contains a particular label, then the label in the boolean vector is set to 1, otherwise, it is set to 0. When a new instance is given to this classifier, it returns one boolean vector with possible label values set to 1. Though it is true that this approach can retrieve the complete label correlations, the upper bound of the combinations are exponential (2^n) and class imbalance issues can occur as the number of instances for each distinct class label may be sparse.

1.2 Aims and Objectives

Label correlation is one of the major criteria which should preserve while building a multi-label classifier yet it can be challenging with the computational complexities.[2] Cost of the classification task could vary from linear to an exponential with the number of possible labels. Not only the training task of the multi-label classification task but also a prediction for a new instance takes extra time as there could be a higher number of annotations to be done. Therefore, the issue of managing the accuracy of multi-label classification while preserving the label correlations has received a lot of attention. Some of the previous studies[3] focused on categorizing label correlation strategies into three main categories as follows.

- The *first-order strategy* : label correlations among labels are totally ignored and consider multi-label classification problem as a number of independent binary classification problems (Ex : BR method).
- The *Second-order strategy* : computes pairwise relations among labels to distinguish between relevant and non-relevant labels or to identify co-occurrences among labels.[4–6]
- The *Higher-order strategies* : considers all possible correlations between all other labels on one label.[2,7]

This study is to incorporate a higher-order strategy for capturing label correlations in a multi-label data set and build multi-label classifiers that preserve the higher-order label correlations. The proposed method, which named as **Multi-Label Classification with Label Clusters (MLC–LC)** first partitions the set of labels into clusters based on how they co-occur in data set. Then employee two matured problem transformation to classify each and every cluster partitions depending on the size of the partition. However, the clustering method used for this study was well-known **Cover-Coefficient Based Clustering Methodology (**C^3M**)** that computes the label clusters by identifying labels that occur in many records in the data set. The co-occurrence pattern of labels is used by C^3M to compute the number of clusters sufficient for the data set as well the initial seeds (centroids) of these clusters. This unique feature of C^3M is exploited to compute label clusters where the label correlations are captured effectively. The number of classifiers in the proposed approach is the same as the number of label clusters generated using the C^3M method, which tends to be significantly smaller than the number of labels.[8] Consequently, the number of classifiers that need to be trained and used to predict new instances are considerably smaller than the total number of labels.

The main contributions of the study are as follows –

- Proposed a simple, yet novel, a multi-label classification called the *MLC–LC* method that first partitions the label set into groups of correlated labels and trains an LP classifier for each label cluster separately.
- The predictive performance of *MLC–LC* method was compared with that of several established MLC methods such as the RAkELd, RAkELo, HOMER, LP, and BR on a range of diverse data sets. Our method achieved superior performance in almost all experimental scenarios for all the data sets.
- *MLC–LC* also outperformed the established MLC methods even for smaller training set sizes.

1.3 Structure of chapters

A review of related work is presented in chapter 2. The multi-label techniques, evaluation and validation measures are discussed in chapter 3. Chapter 4 describes the design and implementation of the experimental environment and results. The dissertation concludes with chapter 5, which summarizes the key findings and discusses avenues for future work.

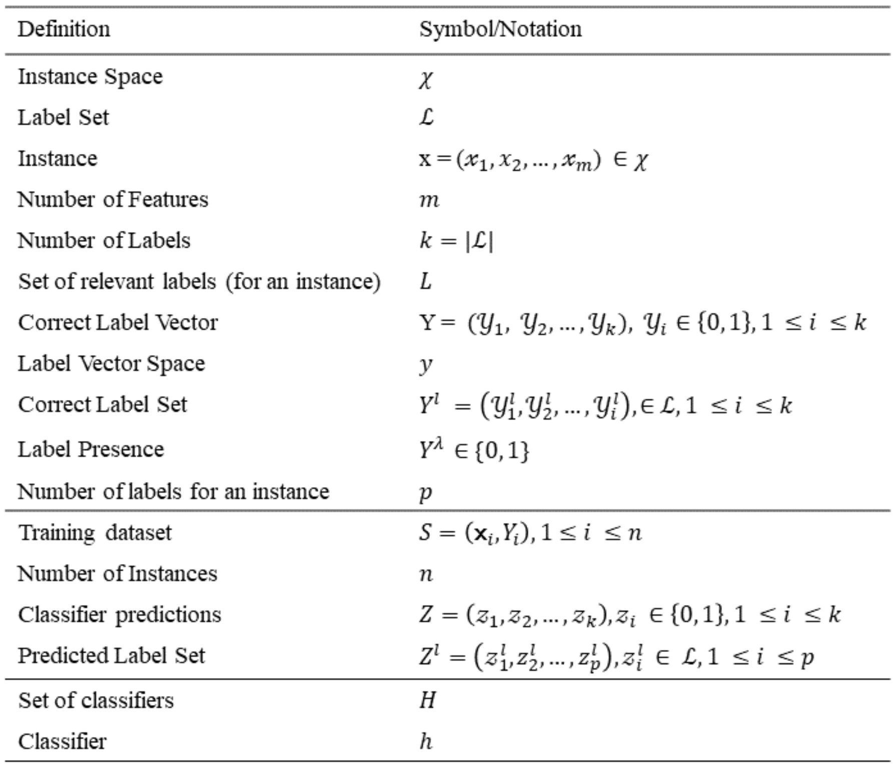
Chapter 2

Literature Review

2.1 Learning Algorithms

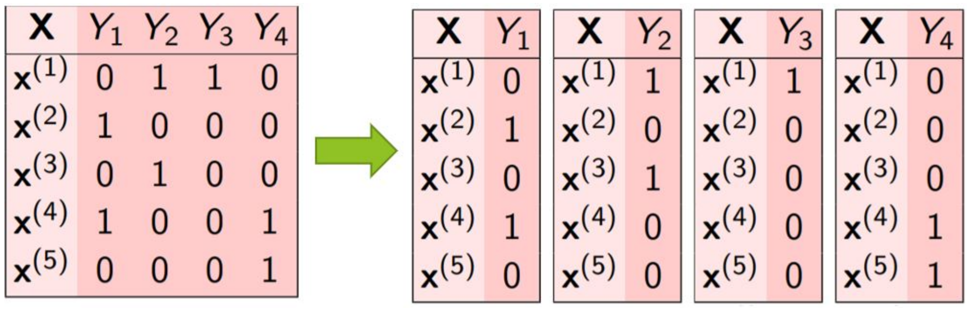
Multi-label classification approaches can be mainly categorized into algorithm adaptation methods and problem transformation methods.[1,9,10] Algorithm adaptation methods extend a specific learning algorithm in order to handle multi-label classification problem directly whereas problem transformation approaches convert a multi-label classification problem into several classification or regression problems such as binary classification or multi-class classification. Several methods have been proposed to incorporate label correlations and feature-label correlations into learning in both algorithm adaptation as well as problem transformation methods.[4,6,10–13] This study focuses on problem-transformation methods in this study since *MLC–LC* is also of the same category.

The Label Power set (LP) method[14] (Figure 2.3)converts multi-label classification problem into a multi-class classification task while preserving label correlations by considering every label combination in the data set. However, it suffers from scalability and sparseness issues. The *pruned problem transformation* method is proposed in[15] alleviate the scalability issues somewhat. The *RAndom k-labEL Set (RAkEL)* approach[7,11,14]



Definition	Symbol/Notation
Instance Space	χ
Label Set	$\mathcal{L}$
Instance	$\mathrm{x} = (x_1, x_2, \dots, x_m) \in \chi$
Number of Features	m
Number of Labels	$k = \lvert\mathcal{L}\rvert$
Set of relevant labels (for an instance)	L
Correct Label Vector	$\mathrm{Y} = (y_1, y_2, \dots, y_k),\ y_i \in \{0,1\}, 1 \le i \le k$
Label Vector Space	y
Correct Label Set	$Y^l = (y_1^l, y_2^l, \dots, y_i^l), \in \mathcal{L}, 1 \le i \le k$
Label Presence	$Y^\lambda \in \{0,1\}$
Number of labels for an instance	p
Training dataset	$S = (\mathbf{x}_i, Y_i), 1 \le i \le n$
Number of Instances	n
Classifier predictions	$Z = (z_1, z_2, \dots, z_k), z_i \in \{0,1\}, 1 \le i \le k$
Predicted Label Set	$Z^l = (z_1^l, z_2^l, \dots, z_p^l), z_i^l \in \mathcal{L}, 1 \le i \le p$
Set of classifiers	H
Classifier	h

Figure 2.1: Symbols and Notations Used in this report



$\mathbf{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	0	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

$\mathbf{X}$	Y_1
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	1
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	0

$\mathbf{X}$	Y_2
$\mathbf{x}^{(1)}$	1
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	1
$\mathbf{x}^{(4)}$	0
$\mathbf{x}^{(5)}$	0

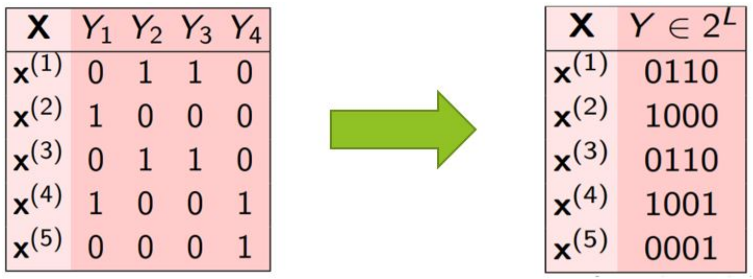
$\mathbf{X}$	Y_3
$\mathbf{x}^{(1)}$	1
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	0
$\mathbf{x}^{(5)}$	0

$\mathbf{X}$	Y_4
$\mathbf{x}^{(1)}$	0
$\mathbf{x}^{(2)}$	0
$\mathbf{x}^{(3)}$	0
$\mathbf{x}^{(4)}$	1
$\mathbf{x}^{(5)}$	1

Figure 2.2: BR method

(Figure 2.4) is one of the popular approaches where the labels are randomly selected to form groups, each containing k labels (k is small compared to the total number of labels) and an LP classifier is trained for each label set. A simple voting process determines the set of labels for each test instance.

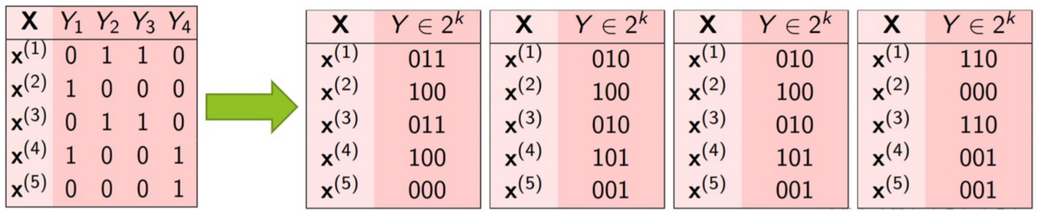
Hullermeier *et al*[16] learn $l(l-1)/2$ binary classifiers, one classifier for each pair of labels in the label set of size l. The data set used to learn each classifier contains instances



$\mathbf{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	1	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

$\mathbf{X}$	$Y \in 2^L$
$\mathbf{x}^{(1)}$	0110
$\mathbf{x}^{(2)}$	1000
$\mathbf{x}^{(3)}$	0110
$\mathbf{x}^{(4)}$	1001
$\mathbf{x}^{(5)}$	0001

Figure 2.3: LP method



$\mathbf{X}$	Y_1	Y_2	Y_3	Y_4
$\mathbf{x}^{(1)}$	0	1	1	0
$\mathbf{x}^{(2)}$	1	0	0	0
$\mathbf{x}^{(3)}$	0	1	1	0
$\mathbf{x}^{(4)}$	1	0	0	1
$\mathbf{x}^{(5)}$	0	0	0	1

$\mathbf{X}$	$Y \in 2^k$	$\mathbf{X}$	$Y \in 2^k$	$\mathbf{X}$	$Y \in 2^k$	$\mathbf{X}$	$Y \in 2^k$
$\mathbf{x}^{(1)}$	011	$\mathbf{x}^{(1)}$	010	$\mathbf{x}^{(1)}$	010	$\mathbf{x}^{(1)}$	110
$\mathbf{x}^{(2)}$	100	$\mathbf{x}^{(2)}$	100	$\mathbf{x}^{(2)}$	100	$\mathbf{x}^{(2)}$	000
$\mathbf{x}^{(3)}$	011	$\mathbf{x}^{(3)}$	010	$\mathbf{x}^{(3)}$	010	$\mathbf{x}^{(3)}$	110
$\mathbf{x}^{(4)}$	100	$\mathbf{x}^{(4)}$	101	$\mathbf{x}^{(4)}$	101	$\mathbf{x}^{(4)}$	001
$\mathbf{x}^{(5)}$	000	$\mathbf{x}^{(5)}$	001	$\mathbf{x}^{(5)}$	001	$\mathbf{x}^{(5)}$	001

Figure 2.4: RAkEL method

for which at least one of the labels in the pair is true, but not both. Furnkranz *et al* in[6] introduced the notion of a *calibration label* that can be used to separate relevant and irrelevant labels predicted by pairwise classification methods to combine multi-class classification with ranking of labels.

There are also multi-label classification methods that incorporate higher-order label dependencies. A classic method is the *classifier chains*[17] that generate l classifiers by incorporating the feature set of the data set given to each classifier by including the label associations assigned to each instance by the previously learned classifiers. Since the performance of a classifier chain may be influenced by the order in which labels are considered by the binary classifiers, *probabilistic* classifier chains[18] have been proposed to predict the best chain ordering. In,[3] authors use association rules to compute the higher order label correlations which are used to select the best training examples. Then, cross-label uncertainty of predicted labels is used to extend the label set of the unseen examples which are then incorporated into the next learning step. Label correlations are also identified to combine predictions from multiple models in[19] to optimize for ranking loss.

The *MLC–LC* methodology is inspired by the *RAkEL* methods that generate groups of labels and learn an LP classifier for each group. The RAkELd method partitions the label set into equal size subsets, each containing k labels for some random integer k whereas RAkELo groups the labels into overlapping subsets. *MLC–LC* also partitions the label set into subsets, however, the number of subsets and the size of each subset are determined by the correlations among labels existing in the data set. The cover-coefficient methodology of C^3M is highly suited for finding the right number of label clusters and the labels that should be grouped into each cluster. The *HOMER*[2] algorithm for multi-label classification also uses the *balanced clustering*, specifically balanced K-means, to distribute a label set into k groups as evenly as possible. Then, it learns a hierarchy of multi-label classifiers, each one learning a smaller label set compared to the classifiers at the previous level. Because of more balanced example distribution, *HOMER* algorithm performed better than the BR method. The label clusters generated by our method *MLC–LC* may not have balanced sizes. In fact, it is common for some of the label clusters to contain single labels and other label clusters may even have a dozen labels, all depending on the data set. Despite this, the results of our experiments show superior predictive performance when compared to that of RAkELd, RAkELo, HOMER, BR, and LP methods.

2.2 Evaluation Matrices

Evaluation matrices are an important component in implementing and maintaining supervised learning model in order to estimate the performances and optimization. Accuracy, the area under the ORC curve (Receiver Operating Characteristic curve), precision and recall are used to evaluate supervised learning models for performances in general. However, evaluating multi-label classification problem is complex than single label classification problem. Multi-label classification performances should be evaluated

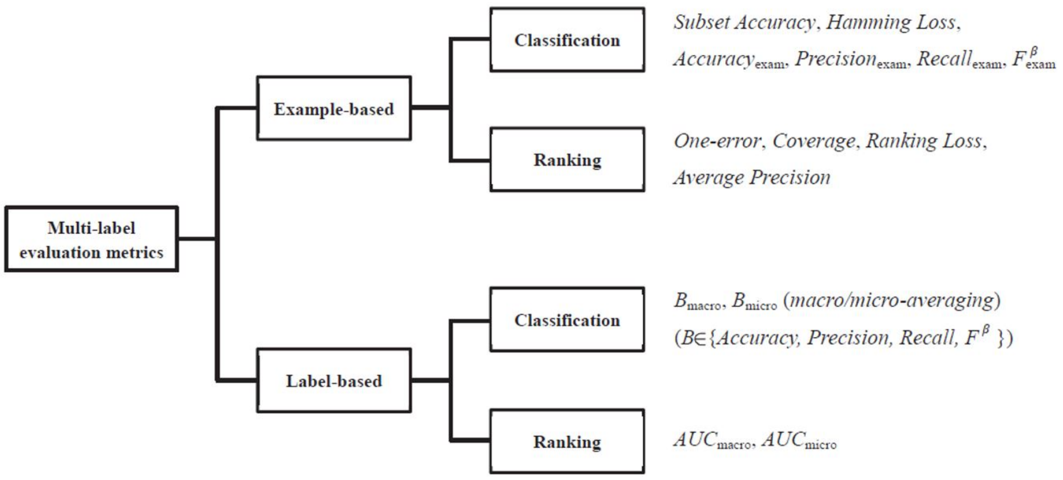

Multi-label evaluation metrics
Example-based
Label-based
Classification
Ranking
Classification
Ranking
Subset Accuracy, Hamming Loss, $Accuracy_{exam}$, $Precision_{exam}$, $Recall_{exam}$, F^{β}_{exam}
One-error, Coverage, Ranking Loss, Average Precision
B_{macro}, B_{micro} (macro/micro-averaging) ($B \in$ {Accuracy, Precision, Recall, F^{β} })
AUC_{macro}, AUC_{micro}

Figure 2.5: Evaluation matrices categorization

both in example wise as well as label wise. Therefore the two main categories of evaluation matrices are employed for multi-label classification. Example based evaluations consider the average prediction difference between actual data in test set where as label based evaluations consider label at a time and average overall label prediction performances (see Figure 2.5).

A classifier may assign all/none/few labels wrongly to a test instance during the prediction phase. Therefore, performance evaluation must take into account each label that was incorrectly predicted. It is also important to consider partial performances during the prediction. To cover most of these aspects, there are many evaluation matrices introduced and most of the evaluation matrices are implemented to capture the correctness/loss in percentage.

2.2.1 Label-based evaluation matrices

- Macro / micro F1-measure

  Recall that L is the set of all labels of a multi-label data set. Let $T = \{(x_i, Z_i)\}$ $(1 \leq i \leq t)$ be a multi-label test set with t-test instances where $Z_i \subseteq L$ is the set of true labels for the i^{th} instance. Let $Y_i \subseteq L$ be the set of labels predicted by a multi-label classifier for the i^{th} instance. The precision of a multi-label classifier computes the ratio of relevant labels predicted by the classifier whereas recall measures the proportion of predicted labels that are relevant. An F_1**-measure** is the harmonic mean of the precision and recall values. It is one of the most frequently used predictive performance-based evaluation metric in the field of classification.[20] The higher the F_1-measure, better the performance.

$$\textit{precision} = \frac{1}{t}\sum_{i=1}^{t}\frac{|Y_i \cap Z_i|}{Z_i}$$
$$\textit{recall} = \frac{1}{t}\sum_{i=1}^{t}\frac{|Y_i \cap Z_i|}{Y_i}.$$

There are many different evaluation metrics for multi-label classification (see[1]). We use the following three metrics, in order to compare our work with other popular methods. (1) *Micro and Macro F_1 scores* : The *micro-F_1* score is the cumulative F_1-measure of over all labels in L whereas the *macro-F_1* score computes the F_1-measure for each label independently and averages these values over $|L|$ to obtain one final measure.[1,10] The formulas for computing the micro-F_1 and macro-F_1 scores are given below. Here, $Z_i^j = 1$ if the i^{th} instance contains label j as one of the true labels, otherwise it is set to 0. Similarly, $Y_i^j = 1$ if label j is predicted as true for the i^{th} instance, otherwise it is set to 0.

$$\textit{micro-}F_1 = \frac{2\sum_{j=1}^{|L|}\sum_{i=1}^{t} Z_i^j Y_i^j}{\sum_{j=1}^{|L|}\sum_{i=1}^{t} Z_i^j + \sum_{j=1}^{|L|}\sum_{i=1}^{t} Y_i^j}$$
$$\textit{macro-}F_1 = \frac{1}{|L|}\sum_{j=1}^{|L|}\frac{2\sum_{i=1}^{t} Z_i^j Y_i^j}{\sum_{j=1}^{|L|}\sum_{i=1}^{t} Z_i^j + \sum_{j=1}^{|L|}\sum_{i=1}^{t} Y_i^j}$$

2.2.2 Example-based evaluation matrices

- *Subset Accuracy*: This matrix evaluates the fraction of exactly correct classified examples. However, subset accuracy evaluation matrix performs poorly when the label set size of the data set is really high as the evaluation criteria are comparatively strict.

$$subsetaccuracy = \frac{1}{t}\sum_{i=1}^{t}\frac{|Y_i = Z_i|}{|L|}$$

- *Hamming Loss (HL)*: Hamming loss is the most commonly used metric to evaluate the performance of a multi-label classifier. It is the size of the average symmetric difference between the set of true labels and the set of predicted labels of a data set. It is computed as follows.

$$HL = \frac{1}{t}\sum_{i=1}^{t}\frac{|Y_i \Delta Z_i|}{|L|}$$

- *One-error*: One-error evaluates the fraction of examples such a way that, top-ranked label is not in the relevant label set.

- *Coverage*: This matrix evaluates how many instances should be travels through on average to cover all the relevant labels if the example.

All the above-mentioned evaluation matrices and other matrices have diverse methods to evaluate predictive and model generalization performances. It is important to select proper evaluation matrices to evaluate the interesting aspects of classification results and performances. However, to make evaluation phrase unbiased and precise, a classification system should be tested with multiple matrices to have a general overview of quality of classification performance from a different perspective.

2.3 Cover Coefficient Clustering method

According to the Information Retrieval System (IRS) concepts, document-based information retrievals mainly related to their terms. Similarity functions are used to determine the relevance of documents. As shown in Figure 2.6 document-term matrix can be built to process the queries. If the element of this matrix is d_{ij} where m number of documents and n number of terms then, $d_{ij} = (1 \leq i \leq m, 1 \leq j \leq n)$, indicates the importance of the term t_j in document d_i. These elements can be either binary values or

$$D = \left(\begin{array}{cccccccc} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \\ d_1 & X & X & X & X & X & X & X \\ d_2 & X & X & X & X & X & X & X \\ d_3 & X & X & X & X & X & X & X \\ d_4 & X & X & X & X & X & X & X \\ d_5 & X & X & X & X & X & X & X \\ d_6 & X & X & X & X & X & X & X \end{array}\right)$$

Figure 2.6: D matrix representing document-term matrix

weighted. As clustering hypothesis is closely associated documents tend to be relevant to the given query. C^3M method belongs to partitioning based clustering approach for document based document retrieval.

This method is first proposed by Fazli-Can et.al[8] to cluster text documents based on word similarities. C^3M method is a partitioning based clustering algorithm which chooses a set of documents as seeds and then all the non-seeds documents are assigned into seed documents so that the set of documents partitioned into clusters lead by seeds. The base concept of C^3M method is Cover Coefficient concept, which serves to identify relationships among documents using document term matrices. Determination of the number of clusters (selecting cluster seeds using cluster seed power) and correlate the documents are done by Cover coefficient concepts. The resultant clustering is guaranteed non-hierarchical clustering and seed based. The Cover Coefficient method determines document relationships of coverage and similarities in multi-dimensional space. C^3M method employees another matrix called C matrix to reflect the above-mentioned coverage and similarities. C matrix is a Document-by-Document matrix which single element, C_{ij} (1 $\leq$ i,j $\leq m$) indicates the probability of selecting any term of d$_i$ from d_j.

C matrix has the information of the relationship between document-based on two-stage probability experiment. This experiment randomly select terms (t) from documents (d) in two stages. In the first stage, if t$_k$ is the term selected randomly of document d_i, then in the second stage, it chooses the selected terms t_k from document

d_j. However, to apply the Cover Coefficient concept, the entries of the D matrix must satisfy the following two conditions. The first condition is no document can be empty which is each document must have at least one term. The second condition is, there should not be any term which does not appear in any document. Further, the C matrix have properties which can be listed as follows,

- $C_{i1} + C_{i2} + C_{i3} + ... + C_{im} = 1$ (All the rows should be sum up to 1)
- For $i \neq j$, 0 $\leq$c$_{ij}$ $\leq$c$_{ii}$ and $c_{ii} > 0$
- If $c_{ij} = 0$, then $cji = 0$ and if $c_{ji} > 0$, then $c_{ji} > 0$.
- If a term of d_i is appeared in another document, then c_{ii} is always less than 1. If not, it is equal to 1.
- $c_{ii} = c_{jj} = c_{ij} = c_{ji}$ iff coupling and decoupling of d_i and d_j are equal.

C^3M method has several characteristics which grab the attention for this study. Some of the characteristics of C^3M are, its capability of determining the number of clusters suitable for a given document set, document distributions within clusters are uniform which ensures moderate cluster sizes (not too large in cluster size or not too large singleton clusters), Cover Coefficient concept guaranteed the independence of the order of the documents clustered in the clustering process. This method is further explained in detail in the methodology section including examples of how it is employed in this study.

Chapter 3

METHODOLOGY

3.1 C^3M Clustering

The proposed approach uses the C^3M clustering method[8] to partition labels into clusters of potentially dependent labels. This section briefly describes the main steps of C^3M method with simple examples below. The input to the algorithm is an $m \times n$ Boolean matrix, E, whose rows are indexed by the set of labels, $L = \{l - 1, \cdots, l_m\}$, and columns are indexed by the elements of set of records, $R = \{r_1, \cdots, r_n\}$. If label l_i is assigned to the record r_u, $E(i, u) = 1$. The *assignment profile* of label l_i is given by the i^{th} row of E and the *labeling profile* of record r_u is given by the u^{th} column of E. The matrix E does not contain any zero columns or zero rows[1] The output of the method is a clustering of labels with the number of clusters being automatically determined by the method. The C^3M method uses a notion of coverage among labels to group them into clusters. The main steps of the C^3M method are given below.

Consider the E matrix in Figure 3.1 with 6 labels and 7 records. Each row specifies the labels that are assigned to the records. For instance, first row of E denotes that test l_1 is

[1]Note that no labels may assigned to a record leading to a zero column corresponding to that record. Such columns are converted to non-zero columns by assigning to an extra "all-zero" label. If a label is not assigned to any of the records, then the corresponding zero row and the label are removed from E and L respectively. More details are provided in the next section.

$$E = \left(\begin{array}{cccccccc} & r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 \\ l_1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ l_2 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ l_3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ l_4 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ l_5 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ l_6 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{array}\right)$$

Figure 3.1: E matrix representing label assignment

$$C = \left(\begin{array}{ccccccc} & l_1 & l_2 & l_3 & l_4 & l_5 & l_6 \\ l_1 & 0.278 & 0.194 & 0.000 & 0.167 & 0.083 & 0.278 \\ l_2 & 0.146 & 0.333 & 0.000 & 0.125 & 0.188 & 0.208 \\ l_3 & 0.000 & 0.000 & 0.500 & 0.000 & 0.500 & 0.000 \\ l_4 & 0.125 & 0.125 & 0.000 & 0.271 & 0.208 & 0.271 \\ l_5 & 0.050 & 0.150 & 0.100 & 0.167 & 0.366 & 0.167 \\ l_6 & 0.167 & 0.167 & 0.000 & 0.216 & 0.167 & 0.283 \end{array}\right)$$

Figure 3.2: The Cover Co-efficient Matrix

assigned to records r_1, r_4, and r_5.

Cover Coefficients: The first step of C^3M method takes E as input and outputs an $m \times m$ square matrix C indexed by the set L. The entries of C denote pairwise *cover coefficients* values among the labels. The *cover coefficient* c_{ij} of a label l_i with respect to a label l_j is the probability that a record r_u labeled by l_i is also labeled by l_j. Informally, the cover coefficient of a label with respect to another denotes the extent to which the assignment profile of the first label is covered by that of the second one. Let α_i and β_u are the reciprocals of the sum of the entries in the i^{th} row and the u^{th} column of the E matrix respectively. The entry c_{ij} in C is obtained using

$$c_{ij} = \alpha_i \times r_{ij} \; where \; r_{ij} = \sum_{k=1}^{n} (E_{ik} \times \beta_k \times E_{jk}). \tag{3.1}$$

Applying Equation (1) to the E matrix of the previous example to obtain the C matrix depicted in Figure 3.2. For instance, $c_{14} = \alpha_1 \times$ [($E_{11} \times \beta_1 \times E_{41}$) + ($E_{12} \times \beta_2 \times E_{42}$) + $\cdots$ + ($E_{17} \times \beta_7 \times E_{47}$)] = $(1/3) \times$ [($1 \times (1/4) \times 1$) + ($0 \times (1/4) \times 1$) + ($0 \times (1/2) \times 0$) + ($1 \times$

$(1/4) \times 1) + (1 \times (1/3) \times 0) + (0 \times (1/3) \times 1) + (0 \times (1/4) \times 0)] = (1/3) \times (1/4 + 1/4) = 0.167$.

As mentioned in Chapter 2, the cover coefficient value c_{ij} can be determined using a two-stage selection process – a) select a record r_k that is assigned the label l_i, and ii) select label l_j from the labels assigned to that record. In Equation (1), the first step of selecting a record r_k that is assigned the label l_i is given by the product $\alpha_i \times E_{ik}$. The second step of arbitrarily selecting the label l_j from all the labels that have been assigned the record r_k is given by the product $\beta_k \times E_{jk}$. To determine the extent to which the assignment profile of l_i is covered by that of l_j, it is needed to consider all the records r_1, $\cdots$, r_n indexing the E matrix and this is given by the sum r_{ij} (The sum r_{ij} is called the *row-covering* of row i by the row j).

Note that the computation of the entry c_{ij} uses assignment profiles of all the labels. In particular, the value of c_{ij} does not equal the ratio of the common number of records assigned both the labels over the total number of records assigned label l_i.
Partitioning Labels into Clusters: The second step of the C^3M method takes the matrix C as input and outputs the number of clusters and the cluster seeds. Each cluster in C^3M consists of a group of labels that are covered maximally by the seed of that cluster. Cluster seeds are labels with distinguishing assignment profiles i.e., they are not likely to be covered by other labels. If a label has a distinguishing assignment profile then it is assigned to records that others are not assigned to, and hence its profile is unlikely to be covered by that of the others. The magnitude of a diagonal entry in the C matrix is used to identify labels with distinguishing profiles. The entry c_{ii} of the i^{th} row is called the *decoupling coefficient* of that row. The decoupling coefficient of the C matrix, δ, is the mean value of the decoupling coefficients of its rows. The number of clusters, n_c, is: $n_c = m \times \delta$. : Continuing, with the running example, the decoupling coefficient for the C matrix in Figure 3.2 is $\delta = (0.278 + 0.333 + 0.500 + 0.271 + 0.366 + 0.283)/6 = 0.339$. The estimated number of clusters is $n_c = 0.339 \times 6 \simeq 3$.

As the sharing among the records assigned a label l_i and assigned other labels decreases, the value of the diagonal entry c_{ii} increases. The entry has a maximum value of 1 when there is no sharing among the records assigned the label l_i and assigned other labels. In other words, if label l_i has a high de-coupling coefficient, then it is not likely to be covered by the other labels and hence is likely to create to its own cluster.It is easy to see that if every label has a high decoupling coefficient, then $n_c = m$ as desired since the decoupling coefficient of the C matrix $\delta = 1$ in this case. It can also be verified that if all the labels have identical profiles, then $n_c = 1$.

The *clustering power* of label l_i is

$$P_i = c_{ii} \times (1 - c_{ii}) \times \sum_{k=1}^{n} E_{ik} \tag{3.2}$$

The labels are ranked based on their clustering power and the top n_c labels are chosen as cluster seeds and are assigned to one cluster each. Ties are broken arbitrarily. A label l_j is considered a *false seed* and eliminated if there exists a seed l_i whose cluster seed power is within a specified threshold δ of that of l_j and the coefficients c_{ii}, c_{jj}, c_{ji} and c_{ij} are sufficiently close, i.e., the magnitude of the pairwise difference of c_{ii}, c_{jj}, c_{ij} and c_{ji} are all within a specified threshold ϵ. In this case, the false seed is eliminated and the next seed in the sorted order is picked. A threshold value of $\epsilon = 0.001$ was used based on the spread of the seed values in the experiments. Each remaining label l_i in L is assigned to a cluster whose seed l_j maximally covers l_i, i.e., c_{ij} is a maximal value, for $1 \leq j \leq n_c$. If more than one seed maximally covers l_i then the label is assigned to the cluster whose maximal covering seed has the higher clustering power (ties are broken arbitrarily). If there exist labels in L that cannot be assigned to any of the clusters because none of the seeds cover them, i.e., $c_{ij} = 0$ for all seed tests t_j then these tests are collected into an additional *ragbag* cluster, [$(n_c + 1)^{th}$ cluster].

The clustering powers for the six labels in our running example are P_1 = 0.602, P_2 = 0.889, P_3 = 0.250, P_4 = 0.790, P_5 = 1.161, and P_6 = 1.015. The seeds of the 3 clusters are l_2, l_5, and l_6. The clusters obtained are C_1 = {l_2}, C_2 = {l_3, l_5}, C_3 = {l_1, l_4, l_6}.

3.2 MLC-LC

Let D be a multi-label data set where D = {(x_i, Z_i) $\mid 1 \leq i \leq n$}, where x_i is a feature vector and Z_i is a subset of a set of labels L = {$l_1, \cdots, l_m$}. First, *MLC–LC* processes D to produce an $m \times n$ Boolean matrix, E, whose rows are indexed by the elements of L and the columns are indexed by the elements of the set of records, R = {$r_1, \cdots, r_n$}. If label l_i $\in Z_u$ and (x_u, Z_u) $\in D$, then E (i, u) = 1, meaning that label l_i is assigned to the record r_u. The *assignment profile* of label l_i is given by the i^{th} row of E and the *labeling profile* of record r_u is given by the u^{th} column of E. We assume that all the assignment and labeling profiles in E to be non-zero, i.e., every record has at least one label assigned to it and every label is assigned to at least one record. Next, the matrix E is input to the C^3M clustering method to partition the label set L into a disjoint set of label clusters $L_1, \cdots,$ L_c.

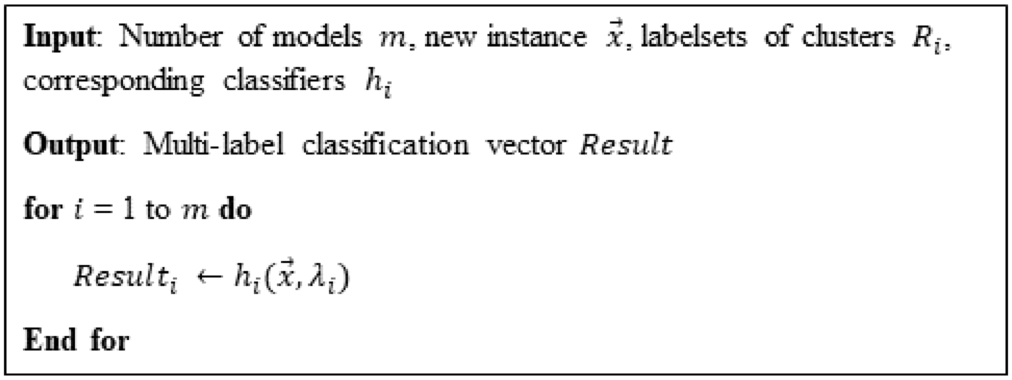
Then, *MLC–LC* learns c classifiers, one for each label cluster. The *MLC–LC* uses the BR method to learn the classifier for label clusters with a single label and uses the LP method otherwise. The training data for a classifier corresponding a singleton label cluster L_k = {l} is obtained from D by replacing each pair (x_i, Z_i) by the pair (x_i, 1) if l belongs to Z_i and is replaced by the pair (x_i, 0) otherwise. If L_k has more than one label, then we add a new label a_l to L_k, the training data is obtained from D by replacing each pair (x_i, Z_i) by the pair (x_i, $Z_i \cap L_k$). In cases where $Z_i \cap L_k$ = {}, (x_i, Z_i) is replaced by (x_i, $\{a_l\}$). It should be clear that the pair $(x_i, \{a_l\})$ is simply a placeholder for a feature vector x_i being assigned all zero values for each of the labels in L_k. Note that *MLC–LC* generates at most $c/2$ new labels. However, this overhead is usually small since the number of clusters $c << m$, the number of labels. Further, extra-label is added to enhance the balance of negative and positive instances in the learning process.

```
**Input**: Set of labels L, training set D
**Output**: Number of models $m$, labelsets of clusters $R_i$, corresponding classifiers $h_i$.
m = c_cube_cluster_result(L)
for $i$=1 to |m| do
    if $|\lambda_i| = 1$ then
        train BR classifier $h_i$ based on $d_i$ and $\lambda_i$
    else
        $d_i = (d_i \cup a)$   $\lhd$ $a$ : balancing column
        train LP classifier $h_i$ based on $d_i$ and $\lambda_i$
    end If
end for
```

Figure 3.3: Non-overlapped label set training process

At the end of the training process, c classifiers are obtained using the training data. Some of these classifiers are BR classifiers and others LP classifiers. A test (or a previously unseen) instance is given to each classifier to get a label assignment. The resulting label assignments, which are disjoint, are simply unioned to obtain an assignment of multiple labels to the test instance. The new labels (i.e; a_l) added while constructing the training set for LP classifiers are removed in the final assignment of labels to the test instances since each a_l is just a place-holder to capture negative instances with respect to a set of labels.

In the process of MLC-LC, clustered label sets (R_i) are classified using most popular problem transformation methods, that is BR and LP methods appropriately. Resultant label clusters use to partition the multi-labeled dataset and run multi-label classification task for individual partitions. Partition sizes are dynamic as the label contained in R_i could be in the range of 1 to L. However, cluster sets have relatively small number of labels $(\lambda << L)$ for most of the multi-label datasets, so that label dimension reduction is done while preserving the label correlations. Hence both BR and LP methods perform efficiently on these smaller partitions of data sets.



```
Input: Number of models $m$, new instance $\vec{x}$, labelsets of clusters $R_i$, corresponding classifiers $h_i$
Output: Multi-label classification vector $Result$
for $i = 1$ to $m$ do
    $Result_i \leftarrow h_i(\vec{x}, \lambda_i)$
End for
```

Figure 3.4: Classification process

In the training process, single labeled clusters ($|\lambda| = 1$) are joined and compute BR method. Clusters with more than one label ($|\lambda| > 1$) train using LP method. Both BR and LP methods use C5.0 decision tree algorithm on training.

3.3 Computational Complexity of MLC–LC

The *MLC–LC* algorithm has two phases – the label clustering phase and the training phase. The time complexity of the C^3M algorithm is $O(m \times x_d \times t_{gs})$ where m is the number of labels, x_d is the average number of distinct records per label, and t_{gs} is the average number of seeds per record.[8] Although x_d is bounded above by n (n is the number of instances in the data set), it is much smaller in practice[2]. Also, the number of seeds is typically much smaller than m and therefore, t_{gs} is also a small value. Therefore, C^3M method can compute label clusters efficiently which was also observed in our experimental set up. Once the label clusters are computed, the training phase learns as many BR/LP classifiers as the number of label clusters. If the complexity of a BR or a LP classifier is $O(g(n))$ where n is the size of the training set, and there are c clusters, then the training phase takes $O(cg(n))$ time to complete.

[2] x_d can be estimated from the label density of a data set as defined in Section 4.1.

Chapter 4

EXPERIMENTAL SETUP

This section contains experimental setup in detail with environment used and experiment results. The main target of this experiment is to compare *MLC–LC* performances with respect to the matured and well-established methods (BR, LP, RAkELd, RAkELo, and HOMER) in the field of multi-label classification. Section 4.1 describes the multi-label data sets, Section 2.2 contains the description of evaluation metrics used, and Section 4.2 is dedicated to results and discussion.

4.1 Data Sets

For these experiments, a variety of data sets from different domains are used. All the data sets used to conduct the experiment are listed in Figure 4.1. The first column, *Dataset,* lists the name of the data set. The *domain* column lists the domain the data collection related to. Columns *Instances, Attributes,* and *Labels* show the number of instances, number of attributes and number of labels respectively in the data set. Another important characteristic of a data set is the proportion of distinct label combinations listed in column *Distinct*. Distinct label combinations capture the complexity of a labeling scheme.

Let D be a multi-label data set consisting of $|D|$ multi-label examples (x_i, Z_i) where x_i is the i^{th} feature vector and $Z_i \subseteq L$ is the set of labels assigned to the i^{th} instance. The

Dataset	Domain	Attributes	Instances	Labels	Distinct	LC	LD
Scene	Image	294	2407	6	15	1.07	0.18
Yeast	Biology	103	2417	14	198	4.237	0.303
Medical	Text	1449	978	45	94	1.245	0.028
Enron	Text	1001	1702	53	753	3.378	0.064
Mediamill	Video	120	43907	101	6555	4.376	0.043
Bibtex	Text	1836	7395	159	2856	2.402	0.015
TMC2007	Text	294	28596	22	1341	2.158	0.098

Table 4.1: Multi-label Data Sets

Label cardinality (*LC*) of D is the average number of labels assigned to the examples in D. It is also known as the standard measure of *multi-labeled-ness*. The *Label density* (*LD*) is the average number of labels of the examples in D divided by $|L|$. The last two columns of the table list these two measures.

$$LC(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} |Z_i|$$
$$LD(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|Z_i|}{|L|}$$

The Scene[21] data set consists of 2407 natural scene images annotated with up to 6 concepts (beach, sunset, field, fall foliage, mountain and urban). Many images contain more than one scene and feature representation is based on spatial color moments of each image. The Yeast[22] data set consists of micro-array expressions and phylogenetic profiles for 2417 yeast genes. Functional classes from the Comprehensive (e.g. metabolism, energy, etc) from the top level of the functional catalog (FunCat) are annotated as 14 labels.

The TMC2007[23] data set is a collection of text data related to aviation safety reports. The original data set contains 28596 safety reports in text form and 22 problem types that appear during flights annotated as labels. Text representation is based on Boolean bag-of-words method. However, in this experiment, the feature set of the TMC2007 data set was reduced to 500 most relevant features for consistent comparison with other methods and manageable computation.

The Medical[24] data set is another text-based data set consisting of documents with free text summaries, patient symptoms histories and prognoses, which were used to predict insurance codes. This data set consists of 978 clinical reports annotated with one or more of 45 disease codes. Enron[25] is another popular text-based multi-label data set which contains collection of email messages exchanged between Enron corporation employees. The number of email messages in the data set is 1702 and each email message is assigned multiple labels from a total of 53 topics.
The Mediamill data set[26] consists of 43907 video frames annotated with 101 labels (e.g. military, desert, basketball, etc). This data collection was part of the Mediamill challenge for automated detection of semantic concept in 2006. Video frames were represented as a set of 120 visual features. The Bibtex data set[27] consists of labels of the Bibtex and Bookmarks corresponds to tags assigned to publications and bookmarks respectively by users of the social bookmark and publication sharing system Bibsonomy. It contains 7395 bibtex entries from the BibSonomy.

As can be observed from Figure 4.1, the size of the label sets ranges from 6 to 159. Although the distinct label combinations are only a fraction of the possible exponential number of label combinations, the number of label combinations is still too high to learn a model for each distinct label combination (In case of Bibtex data set, the number of models needed would be 811). The label density values range from 0.015 for the Bibtex data set to 0.303 for the Yeast data set. This shows that our experiments included data sets where the training data set for a label combination might be sparse, as well as the data sets where the training data set may have a large number of training instances for each label combination.

All the data sets were pre-processed and available in the MLDR R package[20] except the Scene data set that was obtained from Mulan data repository[1], which was used in our experiments.

[1]http://mulan.sourceforge.net/datasets.html

4.2 Results of Experiments

The implementation of *MLC–LC* is mainly done in Java and R statistical programming language and is compared with some of the popular existing multi-label classification approaches. The C^3M algorithm was implemented in Java. The C^3M clustering of the label sets of these sets resulted in 3 clusters for the Yeast data set, 31 clusters for the Medical data set, 15 clusters for the Enron data set, 19 clusters for the Mediamill data set, 9 clusters for the TMC2007 data set, 66 clusters for the Bibtex data set and 6 clusters for the Scene data set. Ragbag cluster was not generated for any of the data sets.

The classification was done in R using the packages – *utiml* (Utilities for Multi-Label Learning)[28] and *mldr* (Exploratory Data Analysis and Manipulation of Multi-Label Data Sets).[20] We compared the performance of *MLC–LC* method to BR, LP, RAkELd, RAkELo, and HOMER methods. All of these methods were available in the *utiml* R package. The RAkELd and RAkELo methods were used with the default setting of $k = 3$ and $m = 2|L|$, the same parameter values used in.[7,11] The C5.0 decision tree learning algorithm was used as the base-level binary classification algorithm for BR, *HOMER*, and the LP classifiers of *RAkEL* and *MLC–LC* in our experiments. For the HOMER method, the default cluster size was set to 3 and the method is set to *balanced*.

All experiments are performed in the Tusker supercomputer cluster hosted by the Holland Computing Center (HCC), configured with 80 GB memory. The evaluation measures are estimated using the holdout cross-validation method using both random sampling and stratified sampling. As the previous studies[28] suggest, feature set normalization and re-scaling are done as dataset preprocessing in order to produce acceptable results.

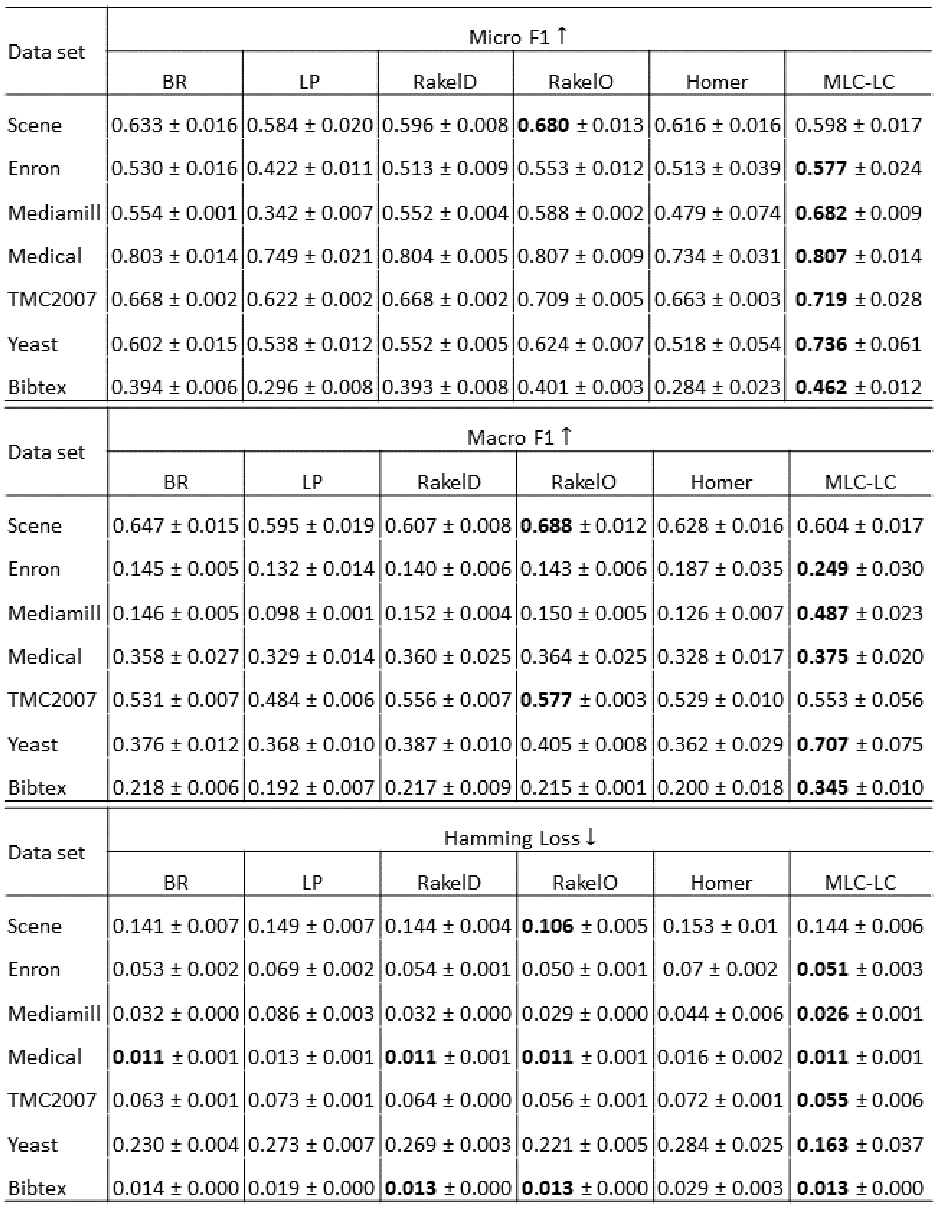
4.2.1 Classification Accuracy of *MLC–LC*

To validate the performance of the proposed method, the experiment was conducted on above mentioned multi-label datasets (Scene, Yeast, Medical, Enron, Mediamill, Bibtex and TMC2007). We compared the classification accuracy of *MLC–LC* to that of some previous methods including BR, LP, RAkELd, RAkELo, and HOMER. Each multi label dataset partitioned into two parts having 66% for the training set and rest for the testing set, according to holdout cross-validation method. All MLC algorithms including *MLC–LC* were trained using the training set and the predictive performance of the models was collected using the test data set. These steps were iterated for 10 times and collected macro-F_1, micro-F_1 scores, and hamming loss values.

For selection of training and testing sets in holdout method, both random sampling selection, as well as stratified sampling selection was employed, where random selection provides blind selection over the dataset and stratified selection divides the dataset into some logical groups (strata) and then samples randomly within those groups. To obtain more representative results, each dataset with each MLC algorithm ran multiple times and averaged evaluation results of macro-F_1, micro-F_1 scores and hamming loss values over the iterations.

Ranking representation mentioned in the previous study[7] used to rank the performances of each MLC algorithm on each dataset. The algorithm that performs the best on a data set (that is the highest micro-F_1 / highest micro-F_1 score / lowest hamming loss) gets a rank of 1, the MLC algorithm with the next best performance gets a rank of 2, etc. Then the average of ranks of each algorithm is calculated over all the data sets. The MLC algorithm with the lowest average rank is considered the best performing MLC algorithm of all the MLC algorithms that were studied in the experiment.

Figure 4.1 presents the average and standard deviation of the micro-F_1, macro-F_1 and hamming loss score measure for all MLC method-dataset pairs, for random sampling whereas 4.2 shows the same for stratified sampling. Overall, *MLC–LC* shows

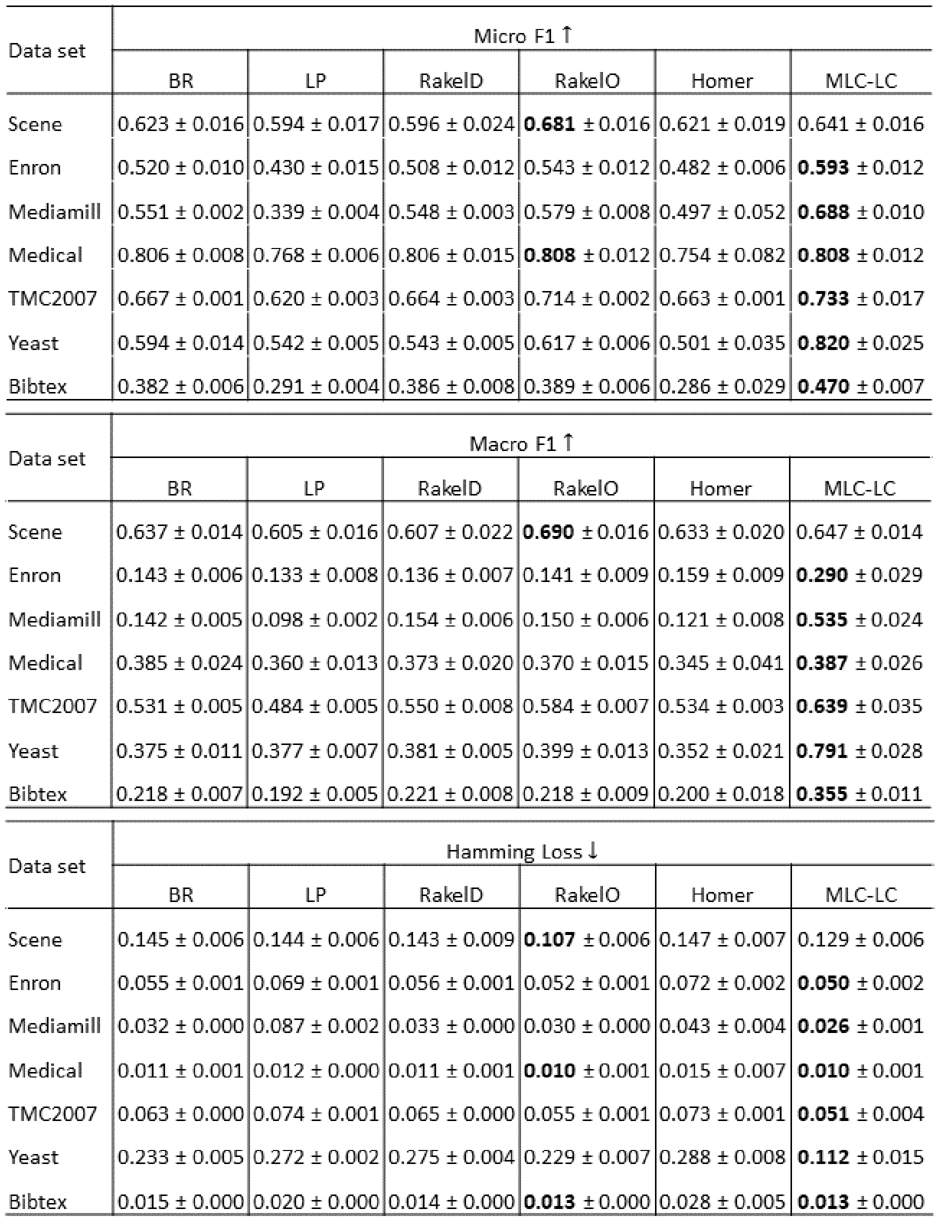


Data set	Micro F1 ↑					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.633 ± 0.016	0.584 ± 0.020	0.596 ± 0.008	**0.680** ± 0.013	0.616 ± 0.016	0.598 ± 0.017
Enron	0.530 ± 0.016	0.422 ± 0.011	0.513 ± 0.009	0.553 ± 0.012	0.513 ± 0.039	**0.577** ± 0.024
Mediamill	0.554 ± 0.001	0.342 ± 0.007	0.552 ± 0.004	0.588 ± 0.002	0.479 ± 0.074	**0.682** ± 0.009
Medical	0.803 ± 0.014	0.749 ± 0.021	0.804 ± 0.005	0.807 ± 0.009	0.734 ± 0.031	**0.807** ± 0.014
TMC2007	0.668 ± 0.002	0.622 ± 0.002	0.668 ± 0.002	0.709 ± 0.005	0.663 ± 0.003	**0.719** ± 0.028
Yeast	0.602 ± 0.015	0.538 ± 0.012	0.552 ± 0.005	0.624 ± 0.007	0.518 ± 0.054	**0.736** ± 0.061
Bibtex	0.394 ± 0.006	0.296 ± 0.008	0.393 ± 0.008	0.401 ± 0.003	0.284 ± 0.023	**0.462** ± 0.012

Data set	Macro F1 ↑					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.647 ± 0.015	0.595 ± 0.019	0.607 ± 0.008	**0.688** ± 0.012	0.628 ± 0.016	0.604 ± 0.017
Enron	0.145 ± 0.005	0.132 ± 0.014	0.140 ± 0.006	0.143 ± 0.006	0.187 ± 0.035	**0.249** ± 0.030
Mediamill	0.146 ± 0.005	0.098 ± 0.001	0.152 ± 0.004	0.150 ± 0.005	0.126 ± 0.007	**0.487** ± 0.023
Medical	0.358 ± 0.027	0.329 ± 0.014	0.360 ± 0.025	0.364 ± 0.025	0.328 ± 0.017	**0.375** ± 0.020
TMC2007	0.531 ± 0.007	0.484 ± 0.006	0.556 ± 0.007	**0.577** ± 0.003	0.529 ± 0.010	0.553 ± 0.056
Yeast	0.376 ± 0.012	0.368 ± 0.010	0.387 ± 0.010	0.405 ± 0.008	0.362 ± 0.029	**0.707** ± 0.075
Bibtex	0.218 ± 0.006	0.192 ± 0.007	0.217 ± 0.009	0.215 ± 0.001	0.200 ± 0.018	**0.345** ± 0.010

Data set	Hamming Loss ↓					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.141 ± 0.007	0.149 ± 0.007	0.144 ± 0.004	**0.106** ± 0.005	0.153 ± 0.01	0.144 ± 0.006
Enron	0.053 ± 0.002	0.069 ± 0.002	0.054 ± 0.001	0.050 ± 0.001	0.07 ± 0.002	**0.051** ± 0.003
Mediamill	0.032 ± 0.000	0.086 ± 0.003	0.032 ± 0.000	0.029 ± 0.000	0.044 ± 0.006	**0.026** ± 0.001
Medical	**0.011** ± 0.001	0.013 ± 0.001	**0.011** ± 0.001	**0.011** ± 0.001	0.016 ± 0.002	**0.011** ± 0.001
TMC2007	0.063 ± 0.001	0.073 ± 0.001	0.064 ± 0.000	0.056 ± 0.001	0.072 ± 0.001	**0.055** ± 0.006
Yeast	0.230 ± 0.004	0.273 ± 0.007	0.269 ± 0.003	0.221 ± 0.005	0.284 ± 0.025	**0.163** ± 0.037
Bibtex	0.014 ± 0.000	0.019 ± 0.000	**0.013** ± 0.000	**0.013** ± 0.000	0.029 ± 0.003	**0.013** ± 0.000

Figure 4.1: micro-F_1, macro-F_1 and hamming Loss scores for 66% random sampling training set. Experimental results (mean$\pm$std) on data. $\uparrow$ ($\downarrow$) indicates the larger (smaller), the better

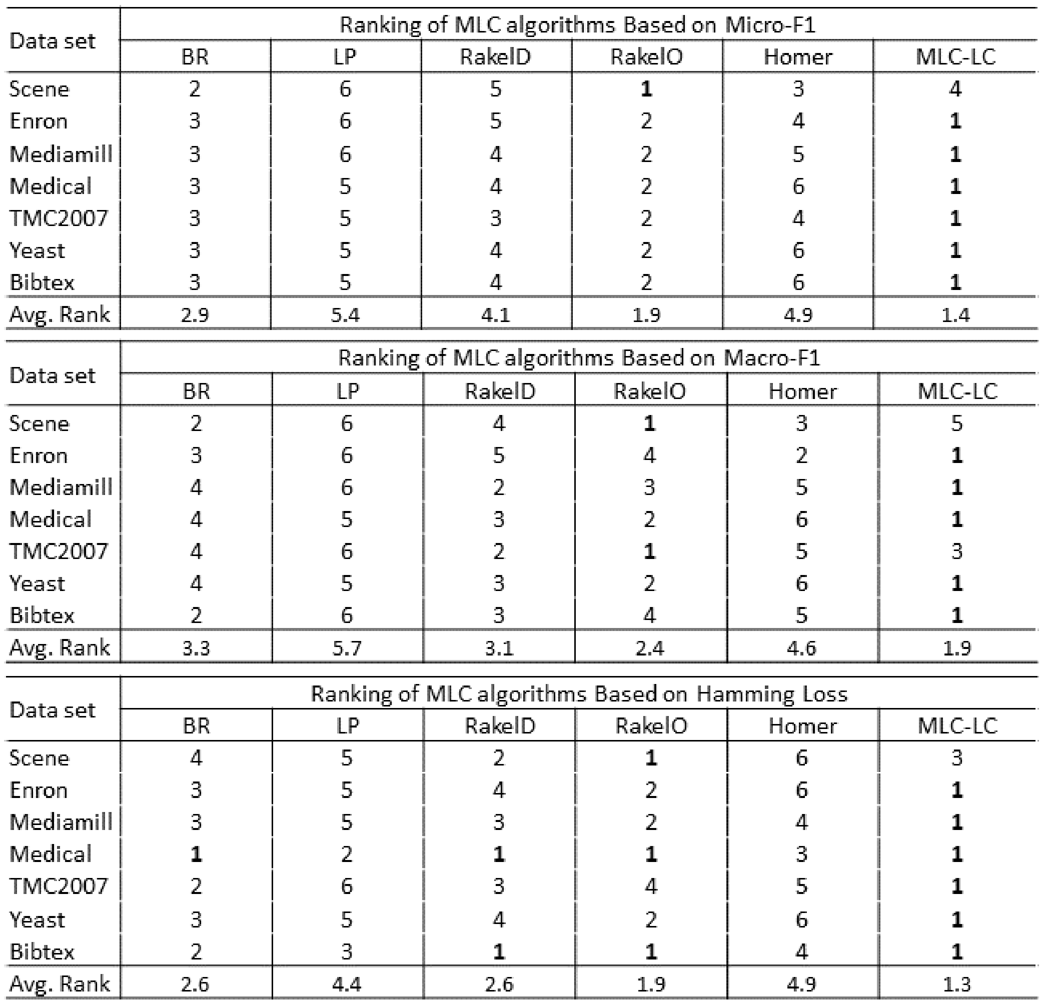


Data set	Micro F1 ↑					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.623 ± 0.016	0.594 ± 0.017	0.596 ± 0.024	**0.681** ± 0.016	0.621 ± 0.019	0.641 ± 0.016
Enron	0.520 ± 0.010	0.430 ± 0.015	0.508 ± 0.012	0.543 ± 0.012	0.482 ± 0.006	**0.593** ± 0.012
Mediamill	0.551 ± 0.002	0.339 ± 0.004	0.548 ± 0.003	0.579 ± 0.008	0.497 ± 0.052	**0.688** ± 0.010
Medical	0.806 ± 0.008	0.768 ± 0.006	0.806 ± 0.015	**0.808** ± 0.012	0.754 ± 0.082	**0.808** ± 0.012
TMC2007	0.667 ± 0.001	0.620 ± 0.003	0.664 ± 0.003	0.714 ± 0.002	0.663 ± 0.001	**0.733** ± 0.017
Yeast	0.594 ± 0.014	0.542 ± 0.005	0.543 ± 0.005	0.617 ± 0.006	0.501 ± 0.035	**0.820** ± 0.025
Bibtex	0.382 ± 0.006	0.291 ± 0.004	0.386 ± 0.008	0.389 ± 0.006	0.286 ± 0.029	**0.470** ± 0.007

Data set	Macro F1 ↑					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.637 ± 0.014	0.605 ± 0.016	0.607 ± 0.022	**0.690** ± 0.016	0.633 ± 0.020	0.647 ± 0.014
Enron	0.143 ± 0.006	0.133 ± 0.008	0.136 ± 0.007	0.141 ± 0.009	0.159 ± 0.009	**0.290** ± 0.029
Mediamill	0.142 ± 0.005	0.098 ± 0.002	0.154 ± 0.006	0.150 ± 0.006	0.121 ± 0.008	**0.535** ± 0.024
Medical	0.385 ± 0.024	0.360 ± 0.013	0.373 ± 0.020	0.370 ± 0.015	0.345 ± 0.041	**0.387** ± 0.026
TMC2007	0.531 ± 0.005	0.484 ± 0.005	0.550 ± 0.008	0.584 ± 0.007	0.534 ± 0.003	**0.639** ± 0.035
Yeast	0.375 ± 0.011	0.377 ± 0.007	0.381 ± 0.005	0.399 ± 0.013	0.352 ± 0.021	**0.791** ± 0.028
Bibtex	0.218 ± 0.007	0.192 ± 0.005	0.221 ± 0.008	0.218 ± 0.009	0.200 ± 0.018	**0.355** ± 0.011

Data set	Hamming Loss ↓					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.145 ± 0.006	0.144 ± 0.006	0.143 ± 0.009	**0.107** ± 0.006	0.147 ± 0.007	0.129 ± 0.006
Enron	0.055 ± 0.001	0.069 ± 0.001	0.056 ± 0.001	0.052 ± 0.001	0.072 ± 0.002	**0.050** ± 0.002
Mediamill	0.032 ± 0.000	0.087 ± 0.002	0.033 ± 0.000	0.030 ± 0.000	0.043 ± 0.004	**0.026** ± 0.001
Medical	0.011 ± 0.001	0.012 ± 0.000	0.011 ± 0.001	**0.010** ± 0.001	0.015 ± 0.007	**0.010** ± 0.001
TMC2007	0.063 ± 0.000	0.074 ± 0.001	0.065 ± 0.000	0.055 ± 0.001	0.073 ± 0.001	**0.051** ± 0.004
Yeast	0.233 ± 0.005	0.272 ± 0.002	0.275 ± 0.004	0.229 ± 0.007	0.288 ± 0.008	**0.112** ± 0.015
Bibtex	0.015 ± 0.000	0.020 ± 0.000	0.014 ± 0.000	**0.013** ± 0.000	0.028 ± 0.005	**0.013** ± 0.000

Figure 4.2: micro-F_1, macro-F_1 and hamming Loss scores for 66% stratified sampling training set. Experimental results (mean$\pm$std) on data. $\uparrow$ ($\downarrow$) indicates the larger (smaller), the better



Data set	Ranking of MLC algorithms Based on Micro-F1					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	2	6	5	**1**	3	4
Enron	3	6	5	2	4	**1**
Mediamill	3	6	4	2	5	**1**
Medical	3	5	4	2	6	**1**
TMC2007	3	5	3	2	4	**1**
Yeast	3	5	4	2	6	**1**
Bibtex	3	5	4	2	6	**1**
Avg. Rank	2.9	5.4	4.1	1.9	4.9	1.4

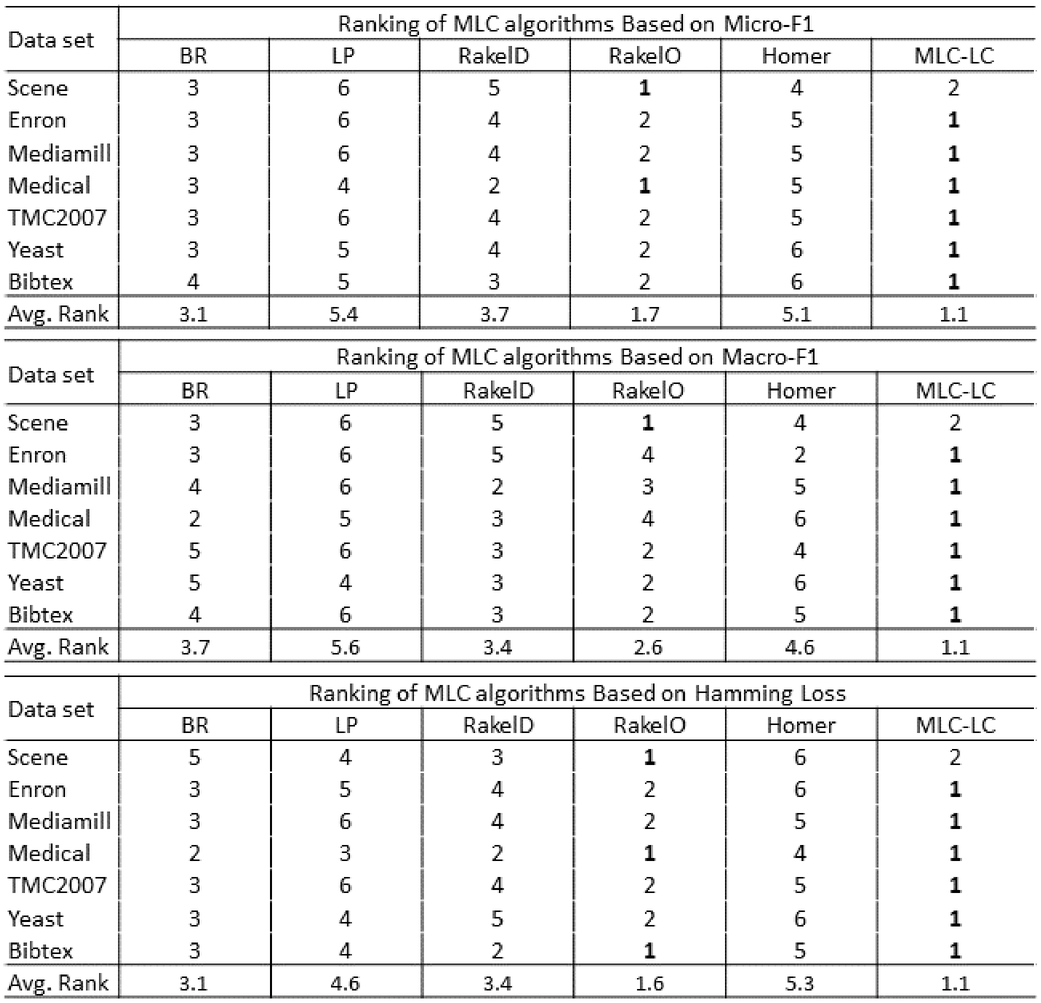
Data set	Ranking of MLC algorithms Based on Macro-F1					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	2	6	4	**1**	3	5
Enron	3	6	5	4	2	**1**
Mediamill	4	6	2	3	5	**1**
Medical	4	5	3	2	6	**1**
TMC2007	4	6	2	**1**	5	3
Yeast	4	5	3	2	6	**1**
Bibtex	2	6	3	4	5	**1**
Avg. Rank	3.3	5.7	3.1	2.4	4.6	1.9

Data set	Ranking of MLC algorithms Based on Hamming Loss					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	4	5	2	**1**	6	3
Enron	3	5	4	2	6	**1**
Mediamill	3	5	3	2	4	**1**
Medical	**1**	2	**1**	**1**	3	**1**
TMC2007	2	6	3	4	5	**1**
Yeast	3	5	4	2	6	**1**
Bibtex	2	3	**1**	**1**	4	**1**
Avg. Rank	2.6	4.4	2.6	1.9	4.9	1.3

Figure 4.3: Ranking of MLC methods according to their micro-F_1, macro-F_1 and hamming loss scores for 66% random sampling training set

comparatively best results for micro-F_1 and macro-F_1 scores over all the other MLC algorithms except for Scene data set. It is same for hamming loss values, as *MLC–LC* shows the lowest loss for both random sampling and stratifies sampling results. For Scene data set, MLC-LC method performs almost same as BR method, since Scene data set is clustered into six singleton clusters by *MLC–LC* clustering approach. As can be seen from these tables, all the evaluation results for stratified sampling are higher than random sampling due to the higher quality of training set selection.

Figures 4.3 and 4.4 show the ranking of different MLC algorithms on different data sets based on evaluation scores for random and stratified sampling respectively. The



Data set	Ranking of MLC algorithms Based on Micro-F1					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	3	6	5	**1**	4	2
Enron	3	6	4	2	5	**1**
Mediamill	3	6	4	2	5	**1**
Medical	3	4	2	**1**	5	**1**
TMC2007	3	6	4	2	5	**1**
Yeast	3	5	4	2	6	**1**
Bibtex	4	5	3	2	6	**1**
Avg. Rank	3.1	5.4	3.7	1.7	5.1	1.1

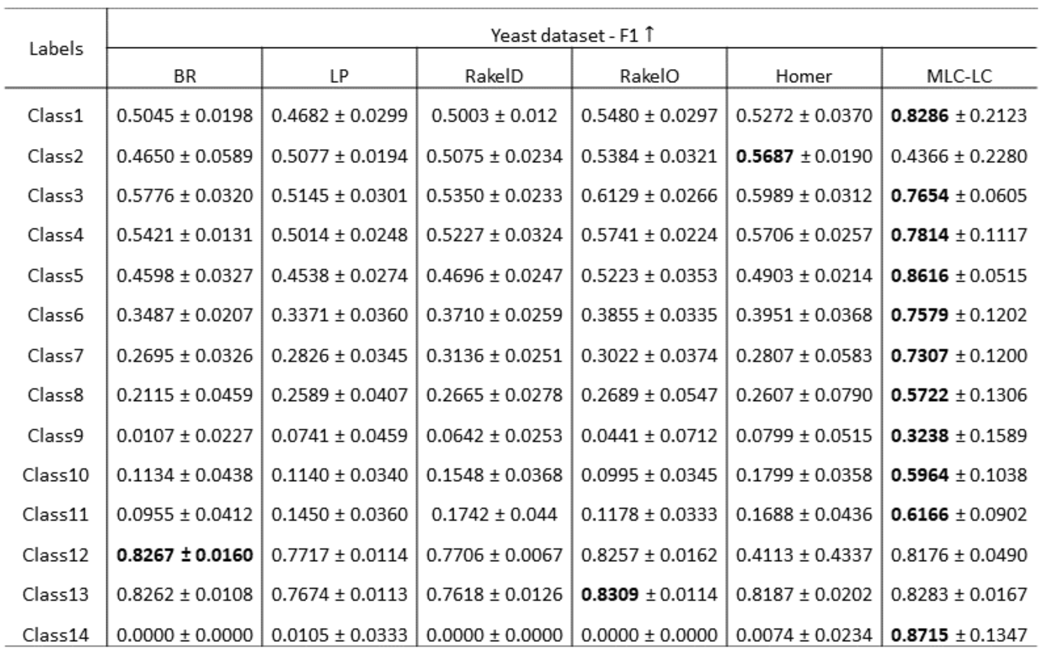
Data set	Ranking of MLC algorithms Based on Macro-F1					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	3	6	5	**1**	4	2
Enron	3	6	5	4	2	**1**
Mediamill	4	6	2	3	5	**1**
Medical	2	5	3	4	6	**1**
TMC2007	5	6	3	2	4	**1**
Yeast	5	4	3	2	6	**1**
Bibtex	4	6	3	2	5	**1**
Avg. Rank	3.7	5.6	3.4	2.6	4.6	1.1

Data set	Ranking of MLC algorithms Based on Hamming Loss					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	5	4	3	**1**	6	2
Enron	3	5	4	2	6	**1**
Mediamill	3	6	4	2	5	**1**
Medical	2	3	2	**1**	4	**1**
TMC2007	3	6	4	2	5	**1**
Yeast	3	4	5	2	6	**1**
Bibtex	3	4	2	**1**	5	**1**
Avg. Rank	3.1	4.6	3.4	1.6	5.3	1.1

Figure 4.4: Ranking of MLC algorithms Based on micro-F_1, macro-F_1 and hamming Loss score for stratified sampling

rows in the tables are the data sets and the columns list the MLC algorithms, except the last row which lists the average rank of each MLC algorithm. For an example, an entry in row i and column j in the first table in 4.3 lists the rank of the micro-F_1 score on the data set i of the micro-F_1 of the MLC algorithm corresponding to j. As can be seen from the *average rank* rows of these two tables, *MLC–LC* has the best performance over all other MLC algorithms and RAkELo has the next best performance.

To explain further insights of *MLC–LC* performances, sub-experiment was conducted on label level evaluation. Figure 4.5 shows the label-wise F_1 score for all the



Labels	Yeast dataset - F1 ↑					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Class1	0.5045 ± 0.0198	0.4682 ± 0.0299	0.5003 ± 0.012	0.5480 ± 0.0297	0.5272 ± 0.0370	**0.8286** ± 0.2123
Class2	0.4650 ± 0.0589	0.5077 ± 0.0194	0.5075 ± 0.0234	0.5384 ± 0.0321	**0.5687** ± 0.0190	0.4366 ± 0.2280
Class3	0.5776 ± 0.0320	0.5145 ± 0.0301	0.5350 ± 0.0233	0.6129 ± 0.0266	0.5989 ± 0.0312	**0.7654** ± 0.0605
Class4	0.5421 ± 0.0131	0.5014 ± 0.0248	0.5227 ± 0.0324	0.5741 ± 0.0224	0.5706 ± 0.0257	**0.7814** ± 0.1117
Class5	0.4598 ± 0.0327	0.4538 ± 0.0274	0.4696 ± 0.0247	0.5223 ± 0.0353	0.4903 ± 0.0214	**0.8616** ± 0.0515
Class6	0.3487 ± 0.0207	0.3371 ± 0.0360	0.3710 ± 0.0259	0.3855 ± 0.0335	0.3951 ± 0.0368	**0.7579** ± 0.1202
Class7	0.2695 ± 0.0326	0.2826 ± 0.0345	0.3136 ± 0.0251	0.3022 ± 0.0374	0.2807 ± 0.0583	**0.7307** ± 0.1200
Class8	0.2115 ± 0.0459	0.2589 ± 0.0407	0.2665 ± 0.0278	0.2689 ± 0.0547	0.2607 ± 0.0790	**0.5722** ± 0.1306
Class9	0.0107 ± 0.0227	0.0741 ± 0.0459	0.0642 ± 0.0253	0.0441 ± 0.0712	0.0799 ± 0.0515	**0.3238** ± 0.1589
Class10	0.1134 ± 0.0438	0.1140 ± 0.0340	0.1548 ± 0.0368	0.0995 ± 0.0345	0.1799 ± 0.0358	**0.5964** ± 0.1038
Class11	0.0955 ± 0.0412	0.1450 ± 0.0360	0.1742 ± 0.044	0.1178 ± 0.0333	0.1688 ± 0.0436	**0.6166** ± 0.0902
Class12	**0.8267 ± 0.0160**	0.7717 ± 0.0114	0.7706 ± 0.0067	0.8257 ± 0.0162	0.4113 ± 0.4337	0.8176 ± 0.0490
Class13	0.8262 ± 0.0108	0.7674 ± 0.0113	0.7618 ± 0.0126	**0.8309** ± 0.0114	0.8187 ± 0.0202	0.8283 ± 0.0167
Class14	0.0000 ± 0.0000	0.0105 ± 0.0333	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0074 ± 0.0234	**0.8715** ± 0.1347

Figure 4.5: F_1 scores for Yeast data set with 66% random sampling training set and 34% testing set

labels in Yeast data set, which shows significantly higher micro-F_1 and macro-F_1 compare to the other MLC algorithms. The superior performance of *MLC–LC* can be clearly seen from this figure as partitioning method uses in *MLC–LC* enhances the predictive performances by categorizing labels by preserving the label dependencies.

Another additional feature introduced in *MLC–LC* is balancing column (a). It is a common characteristic in most of the multi-label datasets that label per instance is low compared to all possible labels in the dataset (Label density). Hence, partitioning the label space into smaller parts introduces a large number of all zero LP label combinations within partitions. This leads to decrease predictive performances for some of the labels with positive bias also known as class imbalance problem. *MLC–LC* algorithm considers about the biasness statistics of the labels within clusters in the training set and determines whether or not to append balancing column into it. This is another reason for higher predictive performances of the proposed method.

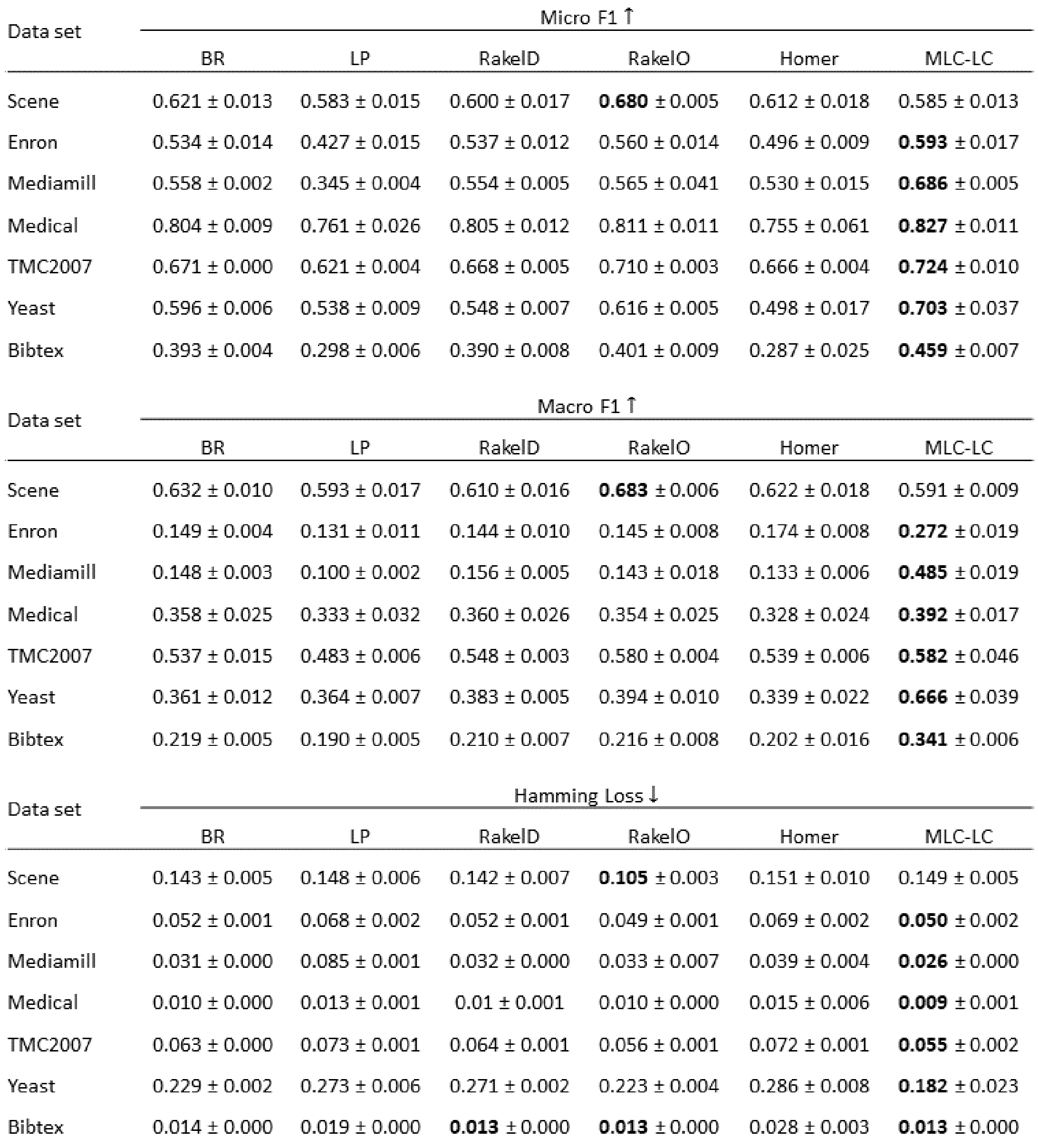
4.2.2 Classification Accuracy with Different Training Set Sizes

This experiment is to illustrate how training set size effects on the predictive performance of MLC-LC compared to the other MLC methods. For this experiment different holdout partitions with 70%, 50%, and 30% of the datasets considered as the training and rest as the testing set. The holdout partitioning is done for both random and stratified as mentioned in the section IV(D). Multi-label classifiers were built using BR, LP, HOMER, RAkELd, RAkELo, and *MLC–LC* methods. The results are evaluated using micro-F_1, macro-F_1 and Hamming Loss matrices. Figure 4.6, Figure 4.7 and Figure 4.8 show the results for 70%, 50%, and 30% training set and testing set selections respectively.

It can be seen that *MLC–LC* method outperforms all the other MLC methods, even with small training set sizes such as 30%. At the same time, it is clear that RAkELo is the only MLC method which performs closer to the *MLC–LC* results. Superior results of MLC-LC obviously hold as the training size increases despite of the diversity of the dataset it tested. We also ranked the relative performances (micro-F_1, macro-F_1 and hamming loss) of each MLC algorithm on each dataset for different training set sizes. According to the average ranking of the performances (given a rank 1, indicates the best performance for a given evaluation score and rank of 2 if it achieves the second highest score, etc), *MLC–LC* method shows smaller value as it consistently outperforms other MLC methods. Hence, in all case, both random and stratified sampling for the training set, *MLC–LC* performed better compared to all the other MLC methods it was compared with.

4.2.3 Classification with Different Label Clustering Algorithms

This experiment compared the performance of an *MLC–LC* classifier using the label clusters obtained from C^3M with those obtained from two popular clustering methods – the K-Means clustering and the hierarchical agglomerative clustering (HAC)[29] . Both

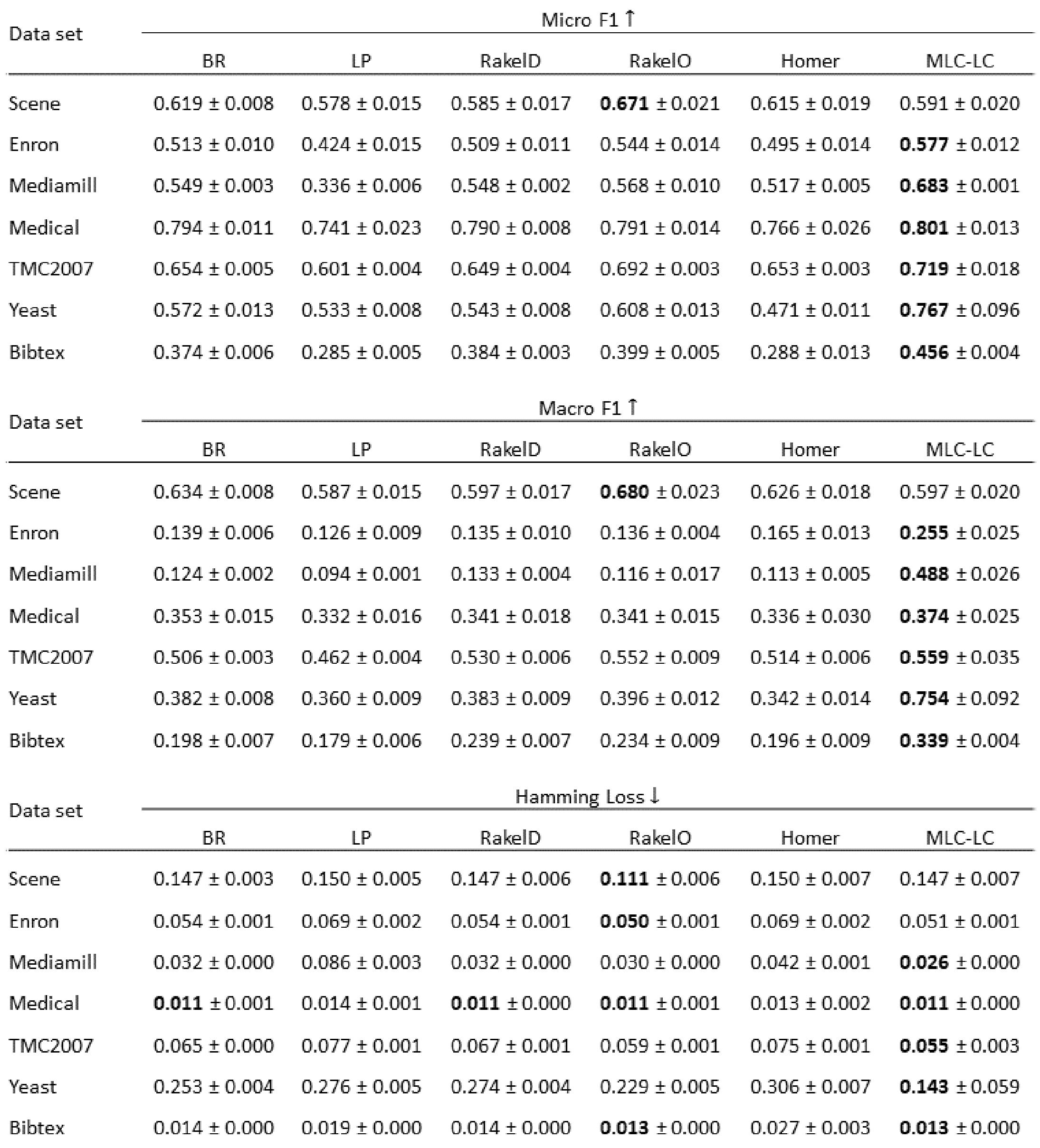


Data set	Micro F1 ↑					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.621 ± 0.013	0.583 ± 0.015	0.600 ± 0.017	**0.680** ± 0.005	0.612 ± 0.018	0.585 ± 0.013
Enron	0.534 ± 0.014	0.427 ± 0.015	0.537 ± 0.012	0.560 ± 0.014	0.496 ± 0.009	**0.593** ± 0.017
Mediamill	0.558 ± 0.002	0.345 ± 0.004	0.554 ± 0.005	0.565 ± 0.041	0.530 ± 0.015	**0.686** ± 0.005
Medical	0.804 ± 0.009	0.761 ± 0.026	0.805 ± 0.012	0.811 ± 0.011	0.755 ± 0.061	**0.827** ± 0.011
TMC2007	0.671 ± 0.000	0.621 ± 0.004	0.668 ± 0.005	0.710 ± 0.003	0.666 ± 0.004	**0.724** ± 0.010
Yeast	0.596 ± 0.006	0.538 ± 0.009	0.548 ± 0.007	0.616 ± 0.005	0.498 ± 0.017	**0.703** ± 0.037
Bibtex	0.393 ± 0.004	0.298 ± 0.006	0.390 ± 0.008	0.401 ± 0.009	0.287 ± 0.025	**0.459** ± 0.007

Data set	Macro F1 ↑					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.632 ± 0.010	0.593 ± 0.017	0.610 ± 0.016	**0.683** ± 0.006	0.622 ± 0.018	0.591 ± 0.009
Enron	0.149 ± 0.004	0.131 ± 0.011	0.144 ± 0.010	0.145 ± 0.008	0.174 ± 0.008	**0.272** ± 0.019
Mediamill	0.148 ± 0.003	0.100 ± 0.002	0.156 ± 0.005	0.143 ± 0.018	0.133 ± 0.006	**0.485** ± 0.019
Medical	0.358 ± 0.025	0.333 ± 0.032	0.360 ± 0.026	0.354 ± 0.025	0.328 ± 0.024	**0.392** ± 0.017
TMC2007	0.537 ± 0.015	0.483 ± 0.006	0.548 ± 0.003	0.580 ± 0.004	0.539 ± 0.006	**0.582** ± 0.046
Yeast	0.361 ± 0.012	0.364 ± 0.007	0.383 ± 0.005	0.394 ± 0.010	0.339 ± 0.022	**0.666** ± 0.039
Bibtex	0.219 ± 0.005	0.190 ± 0.005	0.210 ± 0.007	0.216 ± 0.008	0.202 ± 0.016	**0.341** ± 0.006

Data set	Hamming Loss ↓					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.143 ± 0.005	0.148 ± 0.006	0.142 ± 0.007	**0.105** ± 0.003	0.151 ± 0.010	0.149 ± 0.005
Enron	0.052 ± 0.001	0.068 ± 0.002	0.052 ± 0.001	0.049 ± 0.001	0.069 ± 0.002	**0.050** ± 0.002
Mediamill	0.031 ± 0.000	0.085 ± 0.001	0.032 ± 0.000	0.033 ± 0.007	0.039 ± 0.004	**0.026** ± 0.000
Medical	0.010 ± 0.000	0.013 ± 0.001	0.01 ± 0.001	0.010 ± 0.000	0.015 ± 0.006	**0.009** ± 0.001
TMC2007	0.063 ± 0.000	0.073 ± 0.001	0.064 ± 0.001	0.056 ± 0.001	0.072 ± 0.001	**0.055** ± 0.002
Yeast	0.229 ± 0.002	0.273 ± 0.006	0.271 ± 0.002	0.223 ± 0.004	0.286 ± 0.008	**0.182** ± 0.023
Bibtex	0.014 ± 0.000	0.019 ± 0.000	**0.013** ± 0.000	**0.013** ± 0.000	0.028 ± 0.003	**0.013** ± 0.000

Figure 4.6: micro-F_1, macro-F_1 Scores and hamming loss values of Different MLC methods for Training Set Sizes 70% and Testing set size 30%.

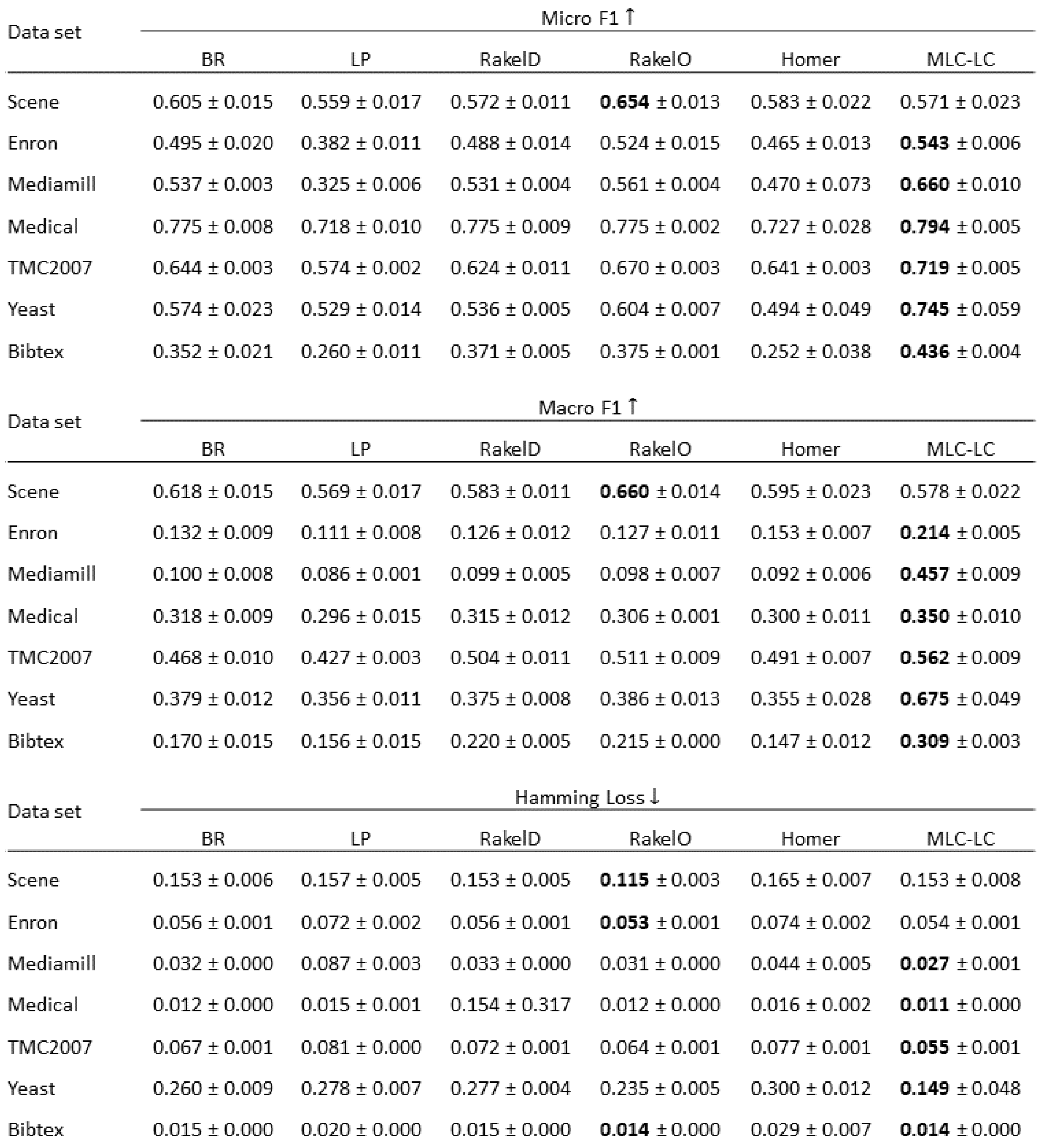


Data set	Micro F1 ↑					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.619 ± 0.008	0.578 ± 0.015	0.585 ± 0.017	**0.671** ± 0.021	0.615 ± 0.019	0.591 ± 0.020
Enron	0.513 ± 0.010	0.424 ± 0.015	0.509 ± 0.011	0.544 ± 0.014	0.495 ± 0.014	**0.577** ± 0.012
Mediamill	0.549 ± 0.003	0.336 ± 0.006	0.548 ± 0.002	0.568 ± 0.010	0.517 ± 0.005	**0.683** ± 0.001
Medical	0.794 ± 0.011	0.741 ± 0.023	0.790 ± 0.008	0.791 ± 0.014	0.766 ± 0.026	**0.801** ± 0.013
TMC2007	0.654 ± 0.005	0.601 ± 0.004	0.649 ± 0.004	0.692 ± 0.003	0.653 ± 0.003	**0.719** ± 0.018
Yeast	0.572 ± 0.013	0.533 ± 0.008	0.543 ± 0.008	0.608 ± 0.013	0.471 ± 0.011	**0.767** ± 0.096
Bibtex	0.374 ± 0.006	0.285 ± 0.005	0.384 ± 0.003	0.399 ± 0.005	0.288 ± 0.013	**0.456** ± 0.004

Data set	Macro F1 ↑					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.634 ± 0.008	0.587 ± 0.015	0.597 ± 0.017	**0.680** ± 0.023	0.626 ± 0.018	0.597 ± 0.020
Enron	0.139 ± 0.006	0.126 ± 0.009	0.135 ± 0.010	0.136 ± 0.004	0.165 ± 0.013	**0.255** ± 0.025
Mediamill	0.124 ± 0.002	0.094 ± 0.001	0.133 ± 0.004	0.116 ± 0.017	0.113 ± 0.005	**0.488** ± 0.026
Medical	0.353 ± 0.015	0.332 ± 0.016	0.341 ± 0.018	0.341 ± 0.015	0.336 ± 0.030	**0.374** ± 0.025
TMC2007	0.506 ± 0.003	0.462 ± 0.004	0.530 ± 0.006	0.552 ± 0.009	0.514 ± 0.006	**0.559** ± 0.035
Yeast	0.382 ± 0.008	0.360 ± 0.009	0.383 ± 0.009	0.396 ± 0.012	0.342 ± 0.014	**0.754** ± 0.092
Bibtex	0.198 ± 0.007	0.179 ± 0.006	0.239 ± 0.007	0.234 ± 0.009	0.196 ± 0.009	**0.339** ± 0.004

Data set	Hamming Loss ↓					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.147 ± 0.003	0.150 ± 0.005	0.147 ± 0.006	**0.111** ± 0.006	0.150 ± 0.007	0.147 ± 0.007
Enron	0.054 ± 0.001	0.069 ± 0.002	0.054 ± 0.001	**0.050** ± 0.001	0.069 ± 0.002	0.051 ± 0.001
Mediamill	0.032 ± 0.000	0.086 ± 0.003	0.032 ± 0.000	0.030 ± 0.000	0.042 ± 0.001	**0.026** ± 0.000
Medical	**0.011** ± 0.001	0.014 ± 0.001	**0.011** ± 0.000	**0.011** ± 0.001	0.013 ± 0.002	**0.011** ± 0.000
TMC2007	0.065 ± 0.000	0.077 ± 0.001	0.067 ± 0.001	0.059 ± 0.001	0.075 ± 0.001	**0.055** ± 0.003
Yeast	0.253 ± 0.004	0.276 ± 0.005	0.274 ± 0.004	0.229 ± 0.005	0.306 ± 0.007	**0.143** ± 0.059
Bibtex	0.014 ± 0.000	0.019 ± 0.000	0.014 ± 0.000	**0.013** ± 0.000	0.027 ± 0.003	**0.013** ± 0.000

Figure 4.7: micro-F_1, macro-F_1 Scores and hamming loss values of Different MLC methods for Training Set Sizes 50% and Testing set size 50%.



Data set	Micro F1 ↑					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.605 ± 0.015	0.559 ± 0.017	0.572 ± 0.011	**0.654** ± 0.013	0.583 ± 0.022	0.571 ± 0.023
Enron	0.495 ± 0.020	0.382 ± 0.011	0.488 ± 0.014	0.524 ± 0.015	0.465 ± 0.013	**0.543** ± 0.006
Mediamill	0.537 ± 0.003	0.325 ± 0.006	0.531 ± 0.004	0.561 ± 0.004	0.470 ± 0.073	**0.660** ± 0.010
Medical	0.775 ± 0.008	0.718 ± 0.010	0.775 ± 0.009	0.775 ± 0.002	0.727 ± 0.028	**0.794** ± 0.005
TMC2007	0.644 ± 0.003	0.574 ± 0.002	0.624 ± 0.011	0.670 ± 0.003	0.641 ± 0.003	**0.719** ± 0.005
Yeast	0.574 ± 0.023	0.529 ± 0.014	0.536 ± 0.005	0.604 ± 0.007	0.494 ± 0.049	**0.745** ± 0.059
Bibtex	0.352 ± 0.021	0.260 ± 0.011	0.371 ± 0.005	0.375 ± 0.001	0.252 ± 0.038	**0.436** ± 0.004

Data set	Macro F1 ↑					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.618 ± 0.015	0.569 ± 0.017	0.583 ± 0.011	**0.660** ± 0.014	0.595 ± 0.023	0.578 ± 0.022
Enron	0.132 ± 0.009	0.111 ± 0.008	0.126 ± 0.012	0.127 ± 0.011	0.153 ± 0.007	**0.214** ± 0.005
Mediamill	0.100 ± 0.008	0.086 ± 0.001	0.099 ± 0.005	0.098 ± 0.007	0.092 ± 0.006	**0.457** ± 0.009
Medical	0.318 ± 0.009	0.296 ± 0.015	0.315 ± 0.012	0.306 ± 0.001	0.300 ± 0.011	**0.350** ± 0.010
TMC2007	0.468 ± 0.010	0.427 ± 0.003	0.504 ± 0.011	0.511 ± 0.009	0.491 ± 0.007	**0.562** ± 0.009
Yeast	0.379 ± 0.012	0.356 ± 0.011	0.375 ± 0.008	0.386 ± 0.013	0.355 ± 0.028	**0.675** ± 0.049
Bibtex	0.170 ± 0.015	0.156 ± 0.015	0.220 ± 0.005	0.215 ± 0.000	0.147 ± 0.012	**0.309** ± 0.003

Data set	Hamming Loss ↓					
	BR	LP	RakelD	RakelO	Homer	MLC-LC
Scene	0.153 ± 0.006	0.157 ± 0.005	0.153 ± 0.005	**0.115** ± 0.003	0.165 ± 0.007	0.153 ± 0.008
Enron	0.056 ± 0.001	0.072 ± 0.002	0.056 ± 0.001	**0.053** ± 0.001	0.074 ± 0.002	0.054 ± 0.001
Mediamill	0.032 ± 0.000	0.087 ± 0.003	0.033 ± 0.000	0.031 ± 0.000	0.044 ± 0.005	**0.027** ± 0.001
Medical	0.012 ± 0.000	0.015 ± 0.001	0.154 ± 0.317	0.012 ± 0.000	0.016 ± 0.002	**0.011** ± 0.000
TMC2007	0.067 ± 0.001	0.081 ± 0.000	0.072 ± 0.001	0.064 ± 0.001	0.077 ± 0.001	**0.055** ± 0.001
Yeast	0.260 ± 0.009	0.278 ± 0.007	0.277 ± 0.004	0.235 ± 0.005	0.300 ± 0.012	**0.149** ± 0.048
Bibtex	0.015 ± 0.000	0.020 ± 0.000	0.015 ± 0.000	**0.014** ± 0.000	0.029 ± 0.007	**0.014** ± 0.000

Figure 4.8: micro-F_1, macro-F_1 scores and hamming loss values of Different MLC methods for Training Set Sizes 30% and Testing set size 70%.

these clustering methods lack a crucial aspect – estimating the number of clusters appropriate for a data set. However, this is a central feature of the C^3M method. It automatically estimates the number of clusters appropriate for a given data set. The number of clusters play a crucial role in the overall performance since they determine the number of classifiers (LP and BR) as well their complexity.

For each data set, we estimated the number of clusters using C^3M method by de-constructing the C^3M estimation component and coupling this with other clustering approaches. The multi-label classifiers generated from these clusters were then compared in terms of accuracy. The C^3M estimation component was combined with K-means($C3K$) and hierarchical agglomerative($C3H$) clustering methods to partition the label set into the specified number of label clusters.

The K-means clustering, given the number of clusters (K) and randomly chosen K labels as centroids (or seeds), iteratively groups labels with the centroids based on a similarity (or a distance) metric. In $C3K$, the estimation component of C^3M provides the parameter value K and the required number of centroids are randomly chosen. We executed the K-means algorithm for 10 iterations, each iteration with different randomly chosen seeds and merged the label clusters obtained from the 10 iterations. We used the Euclidean distance metric to compute the clusters.

Hierarchical agglomerative clustering is a bottom-up clustering method where clusters have sub-clusters. For this clustering method, clustering starts with each single label as a separate cluster. Then for each successive iteration, it merges the closest pair of clusters by satisfying some similarity criteria, until all the data is in one cluster. This method provides three different approaches to manipulate the cluster sizes, namely single linkage, complete linkage, and average linkage.[29] We used the complete linkage method for $C3H$ because it defines the dissimilarity value between two clusters to be the maximum dissimilarity value between any single data point in the first cluster and any single data point in the second cluster. Then, in each stage of the clustering process two clusters with the smallest dissimilarity score among them are combined into one cluster. Clusters are repeatedly merged until the number of clusters matches those estimated by the C^3M algorithm for the given data set. We employed the Euclidean distance to compute the dissimilarity scores.

Table 4.2 shows the statistics of label clusters obtained from the three different methods. This experiment was conducted using the Yeast, Medical, Enron, Mediamill, and tmc2007 data sets. The first column in the table lists the name of the data set, second, third, and fourth columns list the minimum and maximum size of a cluster obtained from C^3M, $C3K$ and $C3H$ clustering methods respectively. The last three columns list the standard deviation in the cluster sizes obtained from C^3M, $C3K$ and $C3H$ clustering methods respectively.

As can be seen from the table, although the number of label clusters for each data set is held constant for the three cluster methods, the actual partitioning of label set into different clusters was different in different methods. If we observe the standard deviation of the cluster sizes (columns named *sdM*, *sdK*, and *sdH*), C^3M has the least value for most data sets. This is because the clusters generated from C^3M contain either many singleton clusters or a small number of larger size clusters. The C^3M algorithm does not attempt to balance the cluster sizes and groups labels completely based on cover-coefficient values and do not typically group unrelated labels into the same cluster. However, the deviation in cluster sizes among the three methods did not have a significant impact on the classification accuracy as discussed below. Perhaps, this is because the labels grouped into the same cluster by the three different methods were largely similar.

Once the label clusters are constructed, the steps outlined in Section 3.2 were repeated and trained a BR or an LP classifier based on the size of a label cluster. We then performed holdout cross-validation with the random sample of 66% for training and rest for testing. Testing results are evaluated using micro-F_1 and macro-F_1 values, which are

Data	M	K	H	sdM	sdK	sdH
Yeast	1,10	4,6	2,8	4.72	1.54	3.1
Medical	1,8	1,15	1,15	1.3	2.5	2.5
Enron	1,22	1,35	1,28	5.5	8.7	9.5
Mediamill	1,25	1,74	1,79	7.0	16.6	17.8
tmc2007	1,11	1,14	1,14	3.3	4.3	4.3

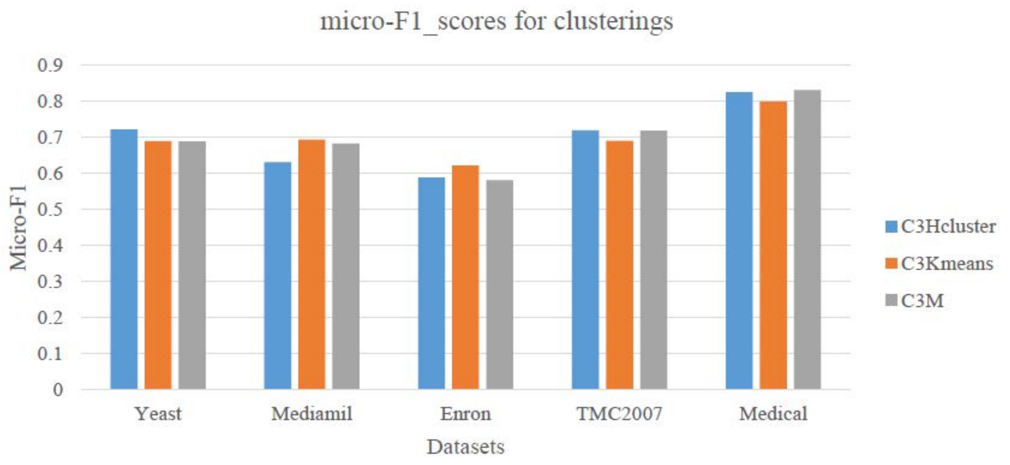
Table 4.2: Label Clustering using C3M, C3K, and C3H methods

displayed in Figure 4.9. From this figure, it can be seen that the predictive performance of the three multi-label classifiers obtained from the three different cluster methods is approximately the same with respect to the micro-F_1 and macro-F_1 measures. The small fluctuations in the micro and macro-F_1 scores are perhaps due to the small variations in the set of labels grouped into the same cluster by the three different clustering methods.

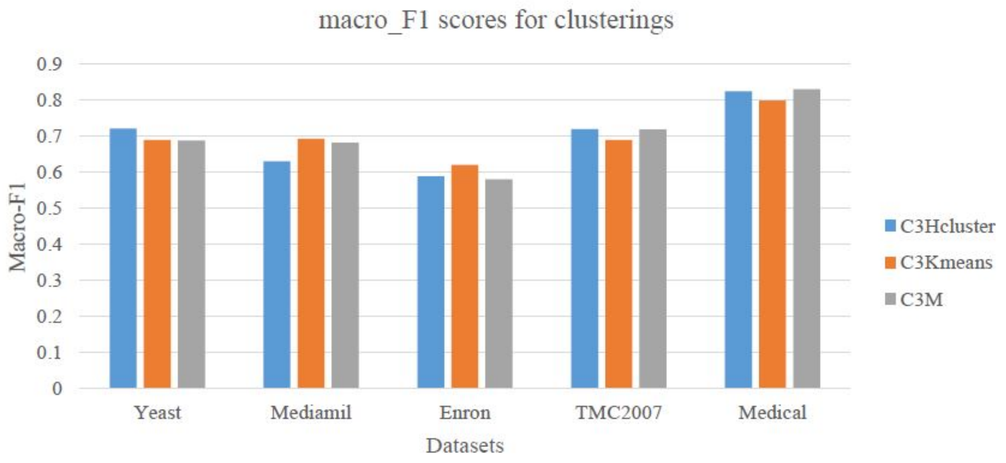
We also computed the Hamming loss of the three multi-label classifiers. We observed that the Hamming loss was small at most 0.189 and as small as 0.009. The Hamming loss of the three multi-label classifiers was almost the same for the Enron, Mediamill, and the tmc2007 data sets. In case of the Yeast and the Medical data sets, the multi-label classifier obtained from $C3H$ had the smallest Hamming loss.

This experiment shows that the choice of clustering method may not affect the predictive performance of the multi-label classifier. However, estimating the appropriate number of clusters for a data set using C^3M can be a powerful tool that can be combined with any clustering algorithm such as the K-means to obtain appropriate partitioning of the data set. Although there are other methods for estimating the number of clusters such as the *gap statistic*[30] method for a given data set, the estimation method used by C^3M analyzes the data dependencies to compute the number of clusters, and hence, possibly more accurate.

In case of sparse label sets, the implementation of C^3M is typically more efficient since it exploits the sparsity in the data. Therefore, one can say that combining the number of clusters estimation from C^3M and then using any other clustering algorithm



(a)



(b)

Figure 4.9: (a) Micro-F_1 and (b) Macro-F_1 scores of the *MLC–LC* Classifiers Obtained from Different Clustering Methods

to find clusters may not be more time-consuming. We are currently studying the cost of executing $C3K$ and $C3H$ on large data sets.

In our experiments, we observed that the RAkELo method was performing either as well as *MLC–LC* or as the next best algorithm on most data sets and experimental settings. This is not surprising as both *MLC–LC* and RAkELo attempt to incorporate label correlations into the classification process. In RAkELo, overlapping subsets of labels are created so that label correlations get appropriate coverage and the voting system ultimately chooses the appropriate label combination for a given instance during the testing phase. In contrast, *MLC–LC* computes the label clusters while preserving the label correlations and the prediction phase is a simple union of predicted labels from multiple LP/BR classifiers.

Chapter 5

CONCLUSIONS AND FUTURE WORK

The multi-label classification methodology which incorporates higher-order label correlations into learning called the *multi-label classification with label clusters (MLC–LC)* proposed in this study is a problem-transformation method. The label set is first partitioned into label clusters depending on how they co-occur in the data set. The training set is constructed from the original data set for each label cluster by including only the labels that occur in the label cluster for each training instance to train a classifier. Therefore, there are as many classifiers as the number of label clusters. Each classifier predicts a set of labels for each test instance. These labels are then unioned to generate the final set of labels. C^3M, a novel clustering algorithm, is used to generate label clusters. Unique features of C^3M include automatically estimating the appropriate number of clusters for a given data set and automatically selecting the cluster seeds. Based on our experimental results, the *MLC–LC* has superior predictive performance over established MLC techniques such as RAkELd, RAkELo, HOMER, etc., on several diverse multi-label data sets. The superior predictive performance of the *MLC–LC* does not suffer even when the training set size is reduced to just 30% of the data set. The classification accuracy of the *MLC–LC* technique when the label clusters are generated is compared using well-known clustering techniques – K-means and complete linkage

HAC. The number of label clusters appropriate for each label set computed by C^3M was provided to these algorithms as an input. It may be because of these reasons that the classification accuracies of the MLC classifiers constructed using label clusters from all three clustering methods were very similar.

In future, we plan to study how to incorporate the effect of feature vector similarity (or dissimilarities) into label correlations. Also, this study can be further improved to accommodate databases with missing labels. Missing labels is one of the major problems that reduces the performance of classification as for some instances are not assigned labels completely. As C^3M method is capable enough to extract label dependencies, it is possible to extend the current study to impute missing labels. Further, we planned to improve the performance of multi-label classification in data streams as another possible future direction since it requires updating the models incrementally. In stream data, it is challenging to maintain label dependencies as new labels may be added or removed as new instances are continuously Incorporated into the analysis.

References

[1] Eva Gibaja and Sebastian Ventura. A tutorial on multi-label learning. *ACM Computing Surveys*, 47, 04 2015.

[2] G. Tsoumakas, I. Katakis, and I. Vlahavas. Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, page XX, 2008.

[3] B. Zhang, Y. Wang, and F. Chen. Multilabel image classification via high-order label correlation driven active learning. *IEEE Transactions on Image Processing*, 23(3):1430–1441, March 2014.

[4] Eyke HÃijllermeier, Johannes FÃijrnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16):1897 – 1916, 2008.

[5] Nadia Ghamrawi and Andrew McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 195–200, New York, NY, USA, 2005. ACM.

[6] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Mach. Learn.*, 73(2):133–153, November 2008.

[7] Grigorios Tsoumakas and Ioannis Vlahavas. *Random k-Labelsets: An Ensemble Method for Multilabel Classification*, pages 406–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[8] Fazli Can and Esen A. Ozkarahan. Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. *ACM Trans. Database Syst.*, 15(4):483–517, December 1990.

[9] M. Zhang and Z. Zhou. A review on multi-label learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, PP(99):1, 2013.

[10] C.X. Li. *Exploiting Label Correlations for Multi-label Classification*. University of California, San Diego, 2011.

[11] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, 99(1), 2010.

[12] Shan Wu, Shangfei Wang, and Qiang Ji. Capturing dependencies among labels and features for multiple emotion tagging of multimedia data. In *AAAI*, 2017.

[13] Shangfei Wang, Jun Wang, Zhaoyu Wang, and Qiang Ji. Enhancing multi-label classification by modeling dependencies among labels. *Pattern Recognition*, 47(10):3405 – 3413, 2014.

[14] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *Int J Data Warehousing and Mining*, 2007:1–13, 2007.

[15] Jesse Read. A pruned problem transformation method for multi-label classification. In *In: Proc. 2008 New Zealand Computer Science Research Student Conference (NZCSRS*, pages 143–150, 2008.

[16] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artif. Intell.*, 172(16-17):1897–1916, November 2008.

[17] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333, 2011.

[18] Krzysztof Dembczynski, Weiwei Cheng, and Eyke HÃijllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*, pages 279–286. Omnipress, 2010.

[19] Sihong Xie, Xiangnan Kong, Jing Gao, Wei Fan, and Philip S. Yu. Multilabel consensus classification. In *ICDM*, pages 1241–1246. IEEE Computer Society, 2013.

[20] Francisco Charte and David Charte. Working with multilabel datasets in R: The mldr package. *The R Journal*, 7(2):149–162, December 2015.

[21] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification, 2004.

[22] AndrÃľ Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *In Advances in Neural Information Processing Systems 14*, pages 681–687. MIT Press, 2001.

[23] A. N. Srivastava and B. Zane-Ulman. Discovering recurring anomalies in text reports regarding complex space systems. In *Aerospace Conference*, pages 3853–3862, 2005.

[24] K. Crammer, M. Dredze, K. Ganchev, P. P. Talukdar, and S. Carroll. Automatic code assignment to medical text. In *Proc. Workshop on Biological, Translational, and Clinical Language Processing, Prague, Czech Republic, BioNLP07*, pages 129–136, 2007.

[25] B. Klimt and Y Yang. The enron corpus: A new dataset for email classification research. In *ECML 2014: European Conference on Machine Learning, 2004. Proceedings*, pages 217 – 226, 2004.

[26] C. G. M. Snoek, M. Worring, J. C. van Gemert, J. M. Geusebroek, and A. W. M. Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proc. 14th ACM International Conference on Multimedia, MULTIMEDIA06*, pages 421–430, 2006.

[27] I. Katakis, G. Tsoumakas, and I. Vlahavas. Multilabel text classification for automated tag suggestion. In *Proc. ECML PKDD08 Discovery Challenge, Antwerp, Belgium*, pages 75–83, 2008.

[28] Adriano Rivolli. Utilities for multi-label learning, 2017.

[29] Lior Rokach and Oded Maimon. *Clustering Methods*, pages 321–352. Springer US, Boston, MA, 2005.

[30] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.