Stacked-MLkNN: A stacking based improvement to Multi-Label k-Nearest Neighbours

Arjun Pakrashi[*] ARJUN.PAKRASHI@UCDCONNECT.IE
Brian Mac Namee BRIAN.MACNAMEE@UCD.IE
Insight Centre for Data Analytics, University College Dublin, Ireland

Editors: Luís Torgo, Bartosz Krawczyk, Paula Branco and Nuno Moniz.

Abstract

Multi-label classification deals with problems where each datapoint can be assigned to more than one class, or label, at the same time. The simplest approach for such problems is to train independent binary classification models for each label and use these models to independently predict a set of relevant labels for a datapoint. MLkNN is an instance-based lazy learning algorithm for multi-label classification that takes this approach. MLkNN, and similar algorithms, however, do not exploit associations which may exist between the set of potential labels. These methods also suffer from imbalance in the frequency of labels in a training dataset. This work attempts to improve the predictions of MLkNN by implementing a two-layer stack-like method, Stacked-MLkNN which exploits the label associations. Experiments show that Stacked-MLkNN produces better predictions than MLkNN and several other state-of-the-art instance-based learning algorithms.

Keywords: multi-label, stacking, instance-based learning

1. Introduction

Multi-label classification problems are those in which a datapoint can be labelled with more than one class simultaneously (Herrera et al., 2016). For example, images can be labelled as containing multiple objects, music can be labelled with more than one genre, or documents can be labelled with multiple topics.

Multi-label classification problems can be formally defined as follows. Let $\mathbf{x}_i$ be a datapoint from a d-dimensional input space $\mathcal{X}$ of real and/or categorical attributes, and let $\mathcal{L} = \{\lambda_1, \lambda_2, \ldots, \lambda_q\}$ be a set of labels. For each datapoint $\mathbf{x}_i$ there is a subset of labels, $\mathcal{L}_i \subseteq \mathcal{L}$, that are *relevant* to that datapoint — for example the objects in an image. On the other hand, $(\mathcal{L} - \mathcal{L}_i)$ is the subset of labels that are called *irrelevant* for the datapoint.

A typical multi-label dataset is defined as $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) | 1 \leq i \leq n\}$, where n is the number of datapoints in the dataset. Here $\mathbf{x}_i = \{x_{i1,} x_{i2,} \ldots, x_{id}\}$ is a vector indicating the i^{th} datapoint and $\mathbf{y}_i = \{y_{i1,} y_{i2,} \ldots, y_{iq}\}$ is a vector of binary values indicating the set of relevant labels, $\mathcal{L}_i$, for the i^{th} datapoint where $y_{ij} = 1$ if $\lambda_j \in \mathcal{L}_i$, and $y_{ij} = 0$ if $\lambda_j \notin \mathcal{L}_i$. Learning a multi-label classification problem involves learning a model or models, which can predict the relevance of every label in $\mathcal{L}$ for a new datapoint $\mathbf{x}_t$.

There are several multi-label classification methods proposed in the literature. The most basic method, *binary relevance* (Boutell et al., 2004), considers each label independently in

[*] Corresponding author. (arjun.pakrashi@insight-centre.org, arjun.pakrashi@ucdconnect.ie)

a set of binary classification problems. Although the method is simple and intuitive, it fails to take advantage of associations that exist between the labels in a dataset which might contribute towards better predictions. More sophisticated multi-label classification approaches do take advantage of these associations.

Approaches to multi-label classification can be broadly divided into two major categories: *problem transformation* and *algorithm adaptation*. Problem transformation methods transform multi-label datasets so as to make them suitable for the application of standard multi-class classification algorithms (binary relevance can be considered as an example within this category). Algorithm adaptation methods adapt or enhance existing multi-class algorithms to work with multi-label datasets.

For the same reasons that they are interesting in multi-class classification scenarios, *instance-based* and *lazy learners* are interesting for multi-label classification scenarios. These include the advantage of delaying computation to query time to allow different hypothesis to be modelled for each query, and the ease with which a model can be updated by simply adding more datapoints to a training set without any need for retraining (Mitchell, 1997). Instance-based learning approaches have been proposed in the multi-label context, for example in Spyromitros et al. (2008); Younes et al. (2008); Cheng and Hullermeier (2009) and Zhang and Zhou (2007).

Multi-label classification problems inherently face the problem of class imbalance as labels are rarely evenly distributed within a dataset and it is quite common to have very rare, or very frequent labels (Charte et al., 2015). The associations that exist between labels can be exploited to improve the predictions and in the process also decrease the negative effects of imbalanced labels. Taking a lazy learning approach is also advantageous as, should any examples of the rare labels be seen in query data, they can be easily exploited by adding them to the training set.

This paper introduces *Stacked-MLkNN*, an improved instance-based lazy approach for multi-label classification problems falling under the problem transformation category. The performance of this new method is compared to other relevant instance-based state-of-the-art approaches, some of which fall into the algorithm adaptation category and while others are from the problem transformation category.

The remainder of the paper is structured as follows. First, some existing instance-based multi-label classification approaches are briefly described in Section 2. Next, Section 3 presents the proposed method, Stacked-MLkNN. In Section 4 the setup of the experiments to compare Stacked-MLkNN to existing and relevant state-of-the-art instance-based multi-label classification methods are described. The results of these experiments are then presented and discussed in Section 5. Finally, some directions for future work are discussed and the paper is concluded in Section 6.

2. Background Information

This section describes current state-of-the-art instance-based learning approaches to multi-label classification: BRkNN and MLkNN which are binary relevance type algorithms, and DMLkNN, IBLR-ML and IBLR-ML+ which are in the algorithm adaptation category and attempt to improve predictions by taking advantage of label associations. The performance of proposed method, Stacked-MLkNN, described in Section 3, will be compared to these state-of-the-art approaches.

2.1. Binary Relevance k-Nearest Neighbours (BRkNN)

The *binary relevance k-nearest neighbours* algorithm proposed by Spyromitros et al. (2008) is a binary relevance approach to multi-label classification that uses a separate k-nearest neighbour (k-NN) model for each label independently. This means running the k-NN process q times, once for each label. Spyromitros et al. (2008) show how efficiency can be gained with respect to execution time by using a shared similarity matrix across the different k-NN processes.

2.2. Multi-Label k-Nearest Neighbours (MLkNN)

Multi-label k-nearest neighbours (MLkNN) (Zhang and Zhou, 2007) was the first lazy approach proposed specifically for multi-label classification. This is also a binary relevance approach which considers each label independently as a binary classification problem. Instead of a standard k-NN method, however, MLkNN uses the *maximum a-posteriori* (MAP) (Kelleher et al., 2015) approach combined with k-NN. First the prior probability of the relevance of a label λ_l is computed from the training dataset. Next, conditional probabilities on the number of neighbours of a datapoint with λ_l as relevant, conditioned on whether or not λ_l is relevant for the datapoint itself, is calculated, again based on the training data. The relevance of λ_l for a new datapoint $\mathbf{x}_t$ is calculated using Bayes rule, utilising these prior and conditional probabilities. This is performed for each label independently. It has been shown that MLkNN performed better than several algorithms.

2.3. Dependent Multi-Label k-Nearest Neighbours (DMLkNN)

Dependent multi-label k-nearest neighbours (DMLkNN) (Younes et al., 2008) modifies MLkNN to take label associations into account. While predicting the relevance of a label λ_l for $\mathbf{x}_t$, along with the label λ_l, DMLkNN utilises all the other labels $\lambda_q \in \mathcal{L} - \{\lambda_l\}$ of the datapoints in the neighbour of the new datapoint $\mathbf{x}_t$. In Younes et al. (2008) it was shown that with respect to two datasets, DMLkNN was able to perform better than MLkNN over several selected values of k.

2.4. Instance Based Learning by Logistic Regression for Multi-Label learning (IBLR-ML)

Instance based learning by logistic regression for multi-label learning (IBLR-ML) by Cheng and Hullermeier (2009) considers the labels of the neighbourhood of the query instance $\mathbf{x}_t$ as features. Then it derives the relevance of a label based on the influences of all of the labels in this feature set using a logistic regression model. Unlike a standard logistic regression model, where the log-odds ratio is modelled by a linear model, IBLR-ML defines the log-odds ratio using an instance-based approach making the log-odds a function of the nearest neighbours of the query datapoint. For a detailed derivation see Cheng and Hullermeier (2009). The method IBLR-ML+ is an extension of the IBLR-ML method which includes additional features to the input space of the regression models, where the additional features can be the original input space features, along with the label predictions. IBLR-ML and IBLR-ML+ were found to perform well in Cheng and Hullermeier (2009) when compared to MLkNN. Although the method is instance-based, the implementation is not strictly lazy, as it requires to train a logistic regression model or a similar process to estimate the optimal values of coefficients as explained in Cheng and Hullermeier (2009).

2.5. Stacked approaches

Stacking is an ensembling approach that uses a first-level ensemble of classifiers to generate a second-level meta-dataset consisting of predictions from the first level, with the target being the classes from the original dataset (Wolpert, 1992). A classifier is used to extract a model from the second-level meta-dataset, to create the overall output from the model.

Previously stacking based approaches to the multi-label context were explored in Tsoumakas et al. (2009); Godbole and Sarawagi (2004), but not in the context of instance-based learners. Tsoumakas et al. (2009) uses multiple folds to generate the stacked layer dataset, which requires training models and predictions, equal to the number of folds, just to generate the second-level dataset. In Godbole and Sarawagi (2004), the second layer also includes the original input space.

3. The Stacked-MLkNN Method

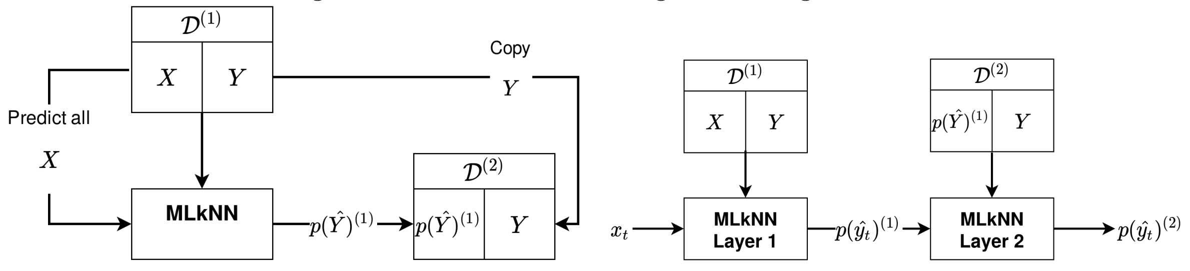
The instance-based learning approaches to multi-label classification MLkNN and BRkNN do not exploit label associations which can lead to better modelling of labels and should be especially advantageous when labels are imbalanced. Although IBLR-ML and IBLR-ML+ do exploit label associations, and are instance-based, they rely on training linear models for each label and therefore cannot be considered strictly lazy approaches. Therefore, adding new datapoints would require retraining these models.

To overcome these problems and improve the performance of MLkNN by taking the label associations into account but still keeping the method lazy, this paper proposes a stack based modification to MLkNN, *Stacked-MLkNN*.

Stacked-MLkNN is a two layer stack-like approach. The first layer predicts the probability of the labels as in MLkNN, whereas the second layer takes the predicted probabilities of each label from the first layer and revises these prediction to take into account associations with predictions for other labels.

The original dataset $\mathcal{D}^{(1)} = \{(\mathbf{x}_i, \mathbf{y}_i) | 1 \leq i \leq n\}$, is used in the first layer. A dataset $\mathcal{D}^{(2)}$ for the second layer is generated from $\mathcal{D}^{(1)}$ using the first layer. For each datapoint $\mathbf{x}_i \in \mathcal{D}^{(1)}$, the predicted probability of relevance, $p(\hat{\mathbf{y}_i})^{(1)}$, for each label is found based on the training set. The second layer dataset is constructed as $\mathcal{D}^{(2)} = \{(p(\hat{\mathbf{y}_i})^{(1)}, \mathbf{y}_i) | 1 \leq i \leq n\}$. Essentially, the predicted probability of the training set based on the training set itself is taken as the input space of the second level. Therefore, in the second level, each target label is made dependent on all the other labels. The second layer also uses MLkNN to perform the final prediction. In the terms of the MLkNN algorithm, for a datapoint $\mathbf{x}_t$, the second layer decides the relevance of a label λ_l based on the how many predictions similar to $p(\mathbf{y}_t^{(1)})$ have the label λ_l as relevant.

Figure 1: Stacked-MLkNN high level diagram



(a) Generate layer 2 dataset $\mathcal{D}^{(2)}$ (b) Prediction process

The dataset $\mathcal{D}^{(2)}$ is generated in a different way from the works mentioned in Section 2.5. Stacked-MLkNN does not use the input space features as described in Godbole and Sarawagi (2004), thus reducing the dimensionality of $\mathcal{D}^{(2)}$. Also, it does not use multiple folds to generate the second layer dataset as in Tsoumakas et al. (2009), therefore saving time to generate $\mathcal{D}^{(2)}$. Therefore the generation of $\mathcal{D}^{(2)}$ takes less time and consists of less attributes, and was found to perform well in the present context.

The prediction for a new datapoint $\mathbf{x}_t$ is performed as follows: The first layer predicts probabilities for each label $p(\hat{\mathbf{y}_t})^{(1)}$ for the datapoint $\mathbf{x}_t$ using MLkNN, based on $\mathcal{D}^{(1)}$. The second layer then uses $p(\hat{\mathbf{y}_t})^{(1)}$ for all λ_l to decide the final label relevance $p(\hat{\mathbf{y}_t})^{(2)}$ based on $\mathcal{D}^{(2)}$ using MLkNN. This process is shown in Figure 1.

As the method uses MLkNN in both the layers, there are two parameters which need to be tuned: k_1 and k_2 the number of nearest neighbours used in MLkNN at the first and second layer of the algorithm respectively.

4. Experiment

To assess the effectiveness of Stacked-MLkNN, experiments were performed using 12 well known multi-label datasets, listed in Table 1. For this experiment all the attributes of each dataset were rescaled between 0 and 1. The columns of the table show the different label properties (Herrera et al., 2016) of the datasets. *Instances*, *Inputs* and *Labels* indicate the total number of data points, the number of predictor variables, and the number of potential labels (q), in each dataset respectively. *Total labelsets* give the number of unique combinations of relevant labels, where each such unique label combination is a *labelset*. *Single Labelsets* indicate the number of data points having a unique combination of relevant labels. *Cardinality* indicates the average number of labels assigned per data point, and *Density* is a normalised dimensionless form of cardinality. *MeanIR* (Herrera et al., 2016) indicates the average degree of label imbalance in the multi-label dataset—a higher value indicates more imbalance.

The implementations for BRkNN, DMLkNN, IBLR-ML and IBLR-ML+ were from the MULAN library (Tsoumakas et al., 2011), whereas MLkNN and the Stacked-MLkNN implementations were by the authors in R[1].

1. Implementation is available at: https://github.com/phoxis/stack_mlknn

Table 1: The datasets used in the experiments described in this paper and their properties

Dataset	Instances	Inputs	Labels	Total Labelsets	Single Labelsets	Cardinality	Density	MeanIR
yeast	2417	103	14	198	77	4.237	0.303	7.197
scene	2407	294	6	15	3	1.074	0.179	1.254
emotions	593	72	6	27	4	1.869	0.311	1.478
medical	978	1449	45	94	33	1.245	0.028	89.501
enron	1702	1001	53	753	573	3.378	0.064	73.953
birds	322	260	20	89	55	1.503	0.075	13.004
genbase	662	1186	27	32	10	1.252	0.046	37.315
cal500	502	68	174	502	502	26.044	0.150	20.578
llog	1460	1004	75	304	189	1.180	0.016	39.267
slashdot	3782	1079	22	156	56	1.181	0.054	17.693
corel5k	5000	499	374	3175	2523	3.522	0.009	189.568
bibtex	7395	1836	159	2856	2199	2.402	0.015	12.498

To measure the performance of the algorithms, the label based *macro averaged F-score* (Zhang and Zhou, 2014) (MAF) is used. This is defined as:

$$F = \frac{1}{q}\sum_{l=1}^{q} 2 \times \frac{Precision_l \times Recall_l}{Precision_l + Recall_l}$$

where $Precision_l$ and $Recall_l$ are the precision and recall scores respectively for the label λ_l, and q is the number of labels. The predicted probabilities were thresholded at 0.5 to determine the label relevance.

In many other studies of multi-label classification (e.g. Spyromitros et al. (2008), Zhang and Zhou (2007), and Cheng and Hullermeier (2009)), *Hamming loss* is used to measure algorithm performance. However, in the presence of imbalanced labels, Hamming loss suffers from the same problems that simple classification accuracy suffers from when used in multi-class problems with imbalanced classes — the performance of models on the majority classes overwhelms performance on minority classes (Kelleher et al., 2015). This is why the macro averaged F-score was chosen over Hamming loss for evaluation.

For each algorithm, a wide range of hyper-parameters were explored to attain the best performance. For each hyper-parameter combination, a 2×5-fold cross-validation was performed for each dataset, and the best mean macro averaged F-score was recorded. The range of hyper-parameter combinations used for each algorithm is described as follows. For BRkNN, MLkNN, IBLR-ML and IBLR-ML+, a total of 12 values of k were explored, where $k \in \{4, 6, 8, \ldots, 26\}$. DMLkNN has two parameters, k and δ, where δ determines the degree of approximate neighbour count as explained in Younes et al. (2008). The hyper-parameter combination explored for DMLkNN was, $(k, \delta) \in \{4, 6, 8, \ldots, 26\} \times \{0, 2, 4, \ldots, 16\}$. Therefore, for this case, 108 hyper-parameter combinations were explored. The smoothing parameter s, for MLkNN and DMLkNN as explained in Zhang and Zhou (2007); Younes et al. (2008), was set to 1.

Stacked-MLkNN has two hyper-parameters, k_1 and k_2, to be selected. Two experiments were performed to select the hyper-parameters. First, a greedy search was performed to first select the k_1 value as the best k value for MLkNN from the set $\{4, 6, 8, \ldots, 26\}$, via cross-validation. With k_1 selected and fixed, the best k_2 value was selected from the set $\{4, 6, 8, \dots, 26\}$, again using cross-validation experiments. Therefore this experiment explored 24 hyper-parameter combinations. As k_1 was first fixed as the best k value for MLkNN, an improvement in the results in Stacked-MLkNN will directly indicate the effectiveness of the stack layer. This tuning was mainly performed to evaluate the effectiveness of the stacked layer and will be discussed in Section 5.

A grid search was also performed over k_1 and k_2 where $(k_1, k_2) \in \{4, 6, 8, \dots, 26\} \times \{4, 6, 8, \dots, 26\}$, with the best parameters selected based on macro-averaged F-score. This process explores a total of 142 hyper-parameter combinations. The main results for Stacked-MLkNN discussed in the next section and presented in Table 2 are based on the hyper parameters chosen via this grid search.

This work isolates the focus on exploiting the label associations in a stacked approach in an instance-based lazy learning context, and therefore does not use kernel methods and/or imbalance reducing pre-processing techniques which may further improve the results.

5. Results

The performance of the Stacked-MLkNN can be assessed in two ways. Firstly, to see if the stack layer did improve the results of the base MLkNN approach. Secondly, to see how it performed compared to the other algorithms mentioned in Section 2.

To show the impact of adding the stacked layer, a greedy approach to find the k_1 and k_2 values was performed as explained in Section 4. Figure 2 shows the macro-averaged F-score for MLkNN as the value of k varies for each dataset. Figure 3 shows the improvement of the Stacked-MLkNN method compared to MLkNN with respect to the values of k_2, when k_1 was fixed as the best value from Figure 2. In Figure 3, the red horizontal line indicates the cross-validated macro-averaged F-score for the best value of k_1 for MLkNN from Figure 2 for the corresponding dataset. The black line shows how the performance changes as the values of k_2 change in the second level of Stacked-MLkNN, when the value of k_1 is fixed to make MLkNN perform the best. For each dataset, the y-axis ranges of Figure 2 and 3 are kept the same, such that a direct visual comparison could be done. Except for the *genbase* dataset, the second level was able to improve the results of MLkNN. As the best parameter for standard MLkNN was used with the first layer of the Stacked-MLkNN method, and then the parameter for the second layer was found, an improvement in the Stacked-MLkNN clearly indicates an improvement over MLkNN's best predictions.

To compare Stacked-MLkNN with the other methods, the values for k_1 and k_2 were selected using a grid search, as explained in Section 4. Table 2 shows the macro averaged F-scores, averaged over the folds for the algorithms mentioned in Section 2 (where a higher value indicates better result). The integer in braces indicates the ranking for that specific dataset over the different algorithms for a specific dataset (where a lower value indicates better rank). The last row shows the average rank for each algorithm over all the datasets. The columns of Table 2 are sorted from best to worst overall average ranking. The values for Stacked-MLkNN shown in the Table 2 are the cross-validated macro averaged F-scores for the best k_1 and k_2 values selected based on the grid search experiment.

From Table 2 it is clear that the Stacked-MLkNN (average rank 2.12) performs better than the standard MLkNN (average rank 4.08). When compared with the other state-of-the-

Table 2: Multi-label macro averaged F-scores illustrating performance of the different algorithms compared in this study across the different datasets used. Integers in brackets indicate relative ranking of the algorithms.

	Stacked-MLkNN	IBLR-ML+	IBLR-ML	MLkNN	BRkNN	DMLkNN
yeast	0.4188 (2)	0.4326 (1)	0.3965 (3)	0.3807 (6)	0.3936 (4)	0.3814 (5)
scene	0.7519 (1.5)	0.6555 (6)	0.7501 (3)	0.7400 (4)	0.6957 (5)	0.7519 (1.5)
emotions	0.6430 (2)	0.6240 (6)	0.6619 (1)	0.6353 (4)	0.6334 (5)	0.6415 (3)
medical	0.5674 (2)	0.5998 (1)	0.4478 (6)	0.5346 (3)	0.4756 (5)	0.5092 (4)
enron	0.2069 (1)	0.1965 (2)	0.1521 (6)	0.1658 (5)	0.1819 (3)	0.1712 (4)
birds	0.3012 (1)	0.2559 (5)	0.2838 (3)	0.2708 (4)	0.2924 (2)	0.2271 (6)
genbase	0.8146 (5)	0.9393 (1)	0.9059 (2)	0.8279 (4)	0.8401 (3)	0.7987 (6)
cal500	0.1252 (4)	0.1827 (1)	0.1766 (2)	0.1022 (6)	0.1335 (3)	0.1143 (5)
llog	0.2719 (1)	0.2239 (5)	0.1115 (6)	0.2599 (2)	0.2573 (4)	0.2585 (3)
slashdot	0.3176 (2)	0.3407 (1)	0.2269 (3)	0.1848 (5)	0.1623 (6)	0.2134 (4)
corel5k	0.2041 (1)	0.1615 (5)	0.0823 (6)	0.1901 (2)	0.1806 (4)	0.1844 (3)
bibtex	0.1416 (3)	0.1453 (2)	0.1607 (1)	0.0747 (4)	0.0602 (5)	0.0415 (6)
Average Rank	2.12	3.00	3.5	4.08	4.08	4.21

art instance-based methods, it can be seen that the Stacked-MLkNN performed better than all others. When compared against the next best approach, IBLR-ML+ (with an average rank of 3.00), Stacked-MLkNN attained rank 1 an equal number of times as IBLR-ML+, but managed to consistently achieve strong ranks when not at the top position.

To assess the degree of difference between the performances of the instance-based methods considered, non-parametric statistical significance tests were performed. Following the recommendation by García et al. (2010), a *Friedman aligned rank test* was performed as the number of algorithms compared is low. The significance test indicated a difference among the algorithms with a significance level of $\alpha = 0.05$. Next, a pairwise post-hoc *Friedman aligned rank test* was performed to explore which algorithms had differed significantly.

The comparisons between algorithms and the p-values from the pairwise post-hoc Friedman aligned rank tests are shown in Table 3 which is partitioned in two halves. A cell in the upper-diagonal of the table indicates the win/lose/tie counts of the algorithm in the corresponding row with respect to the algorithm in the corresponding column. This is included for a direct comparison between the algorithm pairs. For example, Stacked-MLkNN has performed better than MLkNN in 11 of the datasets and worse in 1 dataset. The lower diagonal part of the Table 3 shows the p-values of the post-hoc Friedman aligned rank tests. The asterisk (*) shows if the comparison was found to be significantly different. The different levels of significance (α) are indicated by the number of asterisks, where * indicates $\alpha = 0.10$, ** indicates $\alpha = 0.05$ and *** indicates $\alpha = 0.01$.

It can be seen that Stacked-MLkNN performed better over all datasets than BRkNN and DMLkNN with a significance level of 0.01, and better than MLkNN with a significance level of 0.05. The null hypothesis could be rejected with a significance level of $\alpha = 0.1$ for IBLR-ML, but for IBLR-ML+ the null-hypothesis could not be rejected.

Overall, the results indicate that Stacked-MLkNN was able to perform better than BRkNN, MLkNN and DMLkNN, and it was found not to be very different than IBLR-

Table 3: Significance test
Upper diagonal: win/lose/tie. Lower diagonal: post-hoc Friedman Aligned Rank Test p-values. Significance levels: $* : \alpha = 0.1$, $** : \alpha = 0.05$, $*** : \alpha = 0.01$

	Stacked-MLkNN	IBLR-ML+	IBLR-ML	DMLkNN	MLkNN	BRkNN
Stacked-MLkNN		6/6/0	8/4/0	11/0/1	11/1/0	10/2/0
IBLR-ML+	0.6292		8/4/0	8/4/0	7/5/0	7/5/0
IBLR-ML	* 0.0660	0.1752		7/5/0	8/4/0	7/5/0
DMLkNN	*** 0.0061	** 0.0240	0.3670		6/6/0	6/6/0
MLkNN	** 0.0166	* 0.0559	0.5782	0.7292		7/5/0
BRkNN	*** 0.0083	** 0.0311	0.4238	0.9184	0.8074	

ML and IBLR-ML+, based on this test. Stacked-MLkNN attained the best average rank and is a strictly lazy approach while the other two are not.

It is worth noting that there are non-instance-based and non-lazy learners perform better than the instance-based and lazy learners. For example *classifier chains* (Read et al., 2011) from the problem transformation category, used with support vector machines using radial basis function kernels, and also a neural network based algorithm adaptation method BPMLL (Zhang and Zhou, 2006). However, as these are not instance-based lazy approaches, and do not have the advantages associated with lazy learning, they are not included in this study. A comparison of various multi-label methods including several instance-based methods can be found in Pakrashi et al. (2016).

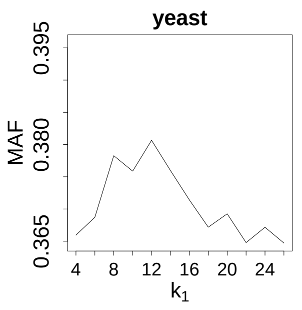
6. Summary and Conclusion

In this paper a stack-like two-layer approach to improve MLkNN for multi-label classification, Stacked-MLkNN, was proposed. The second layer of Stacked-MLkNN predicts the relevance of a label based of the independent label relevance probabilities predicted by MLkNN in the first layer, hence taking the label associations into account. The second layer dataset generation is fast and simple although found to be effective, as the method was able to improve the best results of MLkNN for every dataset (except one). Statistical significance tests indicate that the proposed method was able to perform significantly better than MLkNN and BRkNN which are binary relevance based methods, and also DMLkNN. Stacked-MLkNN seems to also be better than IBLR-ML, but only marginally, where its performance was similar to IBLR-ML+. IBLR-ML and IBLR-ML+ are instance-based, but not strictly lazy approaches, however, and so Stacked-MLkNN has this advantage over them. Given the instance-based nature of Stacked-MLkNN, updating it with new datapoints would be straight forward. This requires adding the new datapoints in the first layer, and the predictions of the first layer of these new datapoints in the second layer dataset.

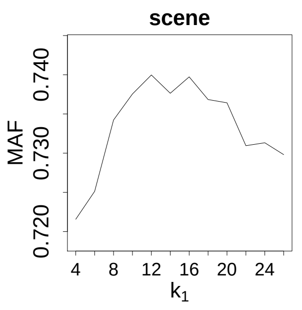
It also should be noted that there are other non-lazy and non-instance-based algorithms which perform better in general than the lazy variants. However, this is seen as an indication of the room left for improvement in instance-based lazy approaches.

The experiments isolate the focus on the effectiveness of the stacked layer of Stacked-MLkNN, which exploits the label associations in an instance-based lazy learning context.

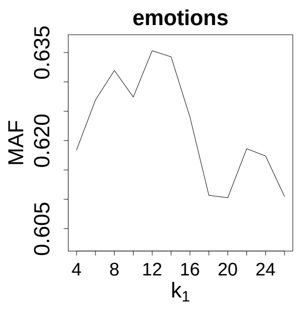
Figure 2: k vs MAF for MLkNN



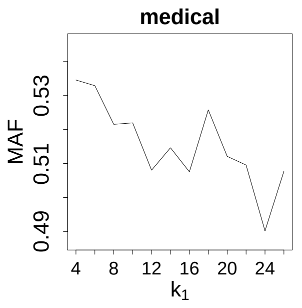
(*a*)



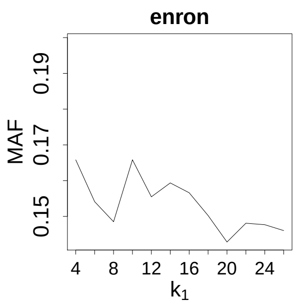
(*b*)



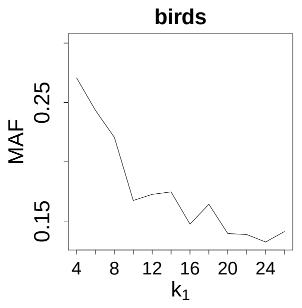
(*c*)



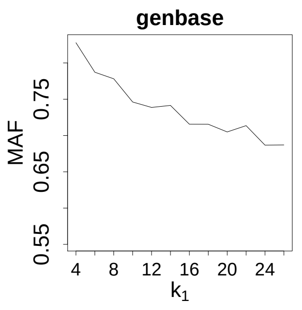
(*d*)



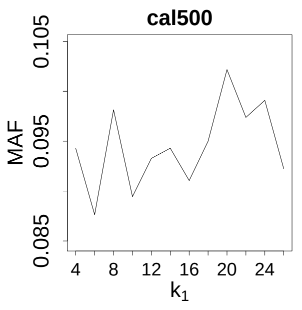
(*e*)



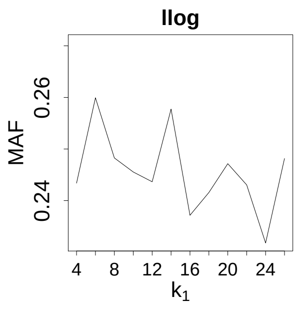
(*f*)



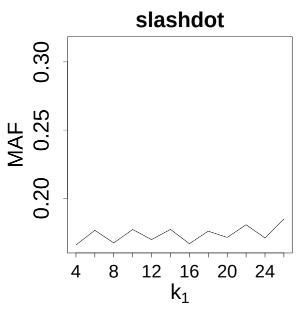
(*g*)



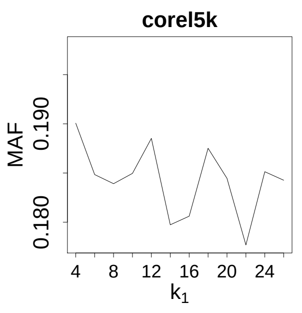
(*h*)



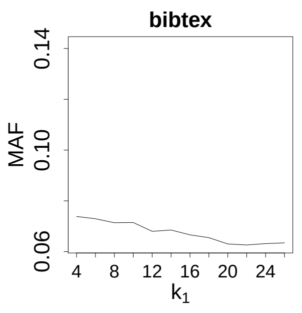
(*i*)



(*j*)

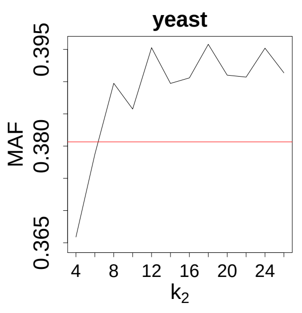


(*k*)

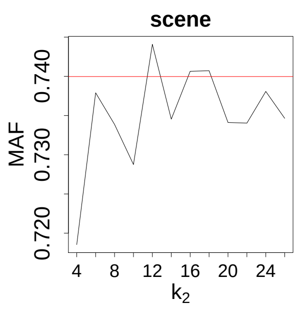


(*l*)

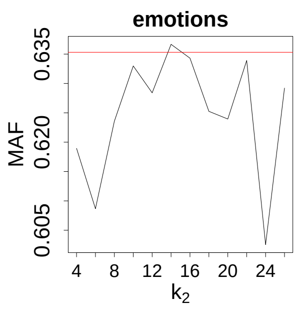
Figure 3: k_2 vs MAF for second layer of Stacked-MLkNN. The red line indicates the best MAF value for MLkNN from Figure 2



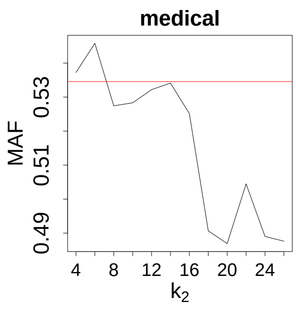
(a)



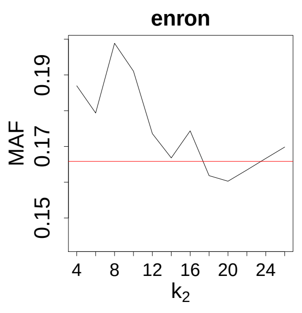
(b)



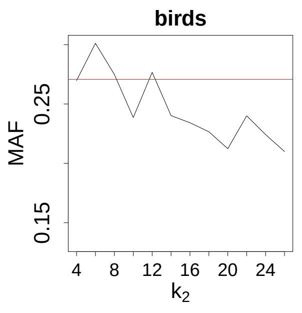
(c)



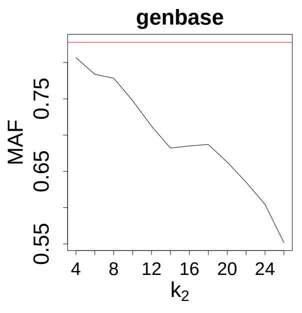
(d)



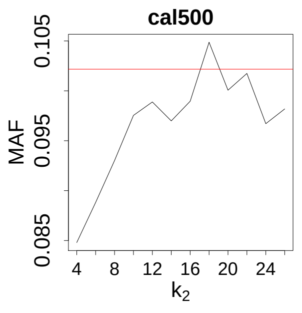
(e)



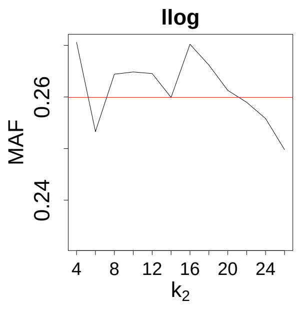
(f)



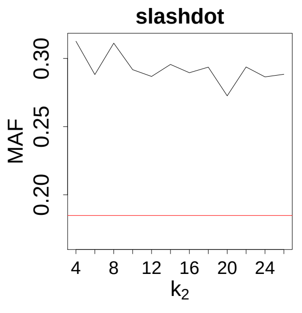
(g)



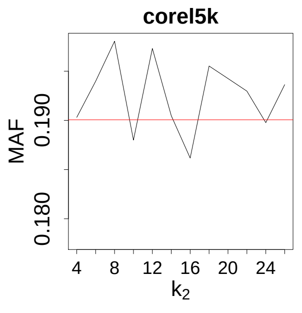
(h)



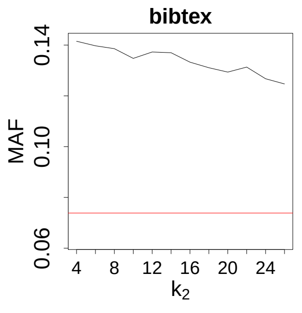
(i)



(j)



(k)



(l)

The results in Section 5 indicate that the Stacked-MLkNN did improve MLkNN and therefore it would now be interesting to further explore Stacked-MLkNN with the focus of improving the predictions.

The results motivate the authors to extend the method towards better prediction quality by investigating the effects of the inclusion of a subset of the input space features along with the labels in the second layer of Stacked-MLkNN; using kernels to improve the prediction quality and also exploring pre-processing methods to improve the label imbalance, while still keeping the process instance-based and lazy.

Acknowledgments

This research was supported by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

References

Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757 – 1771, 2004. ISSN 0031-3203.

Francisco Charte, Antonio J. Rivera, María J. del Jesus, and Francisco Herrera. Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, 163:3 – 16, 2015.

Weiwei Cheng and Eyke Hullermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009. ISSN 0885-6125.

Salvador García, Alberto Fernández, Julián Luengo, and Francisco Herrera. Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180 (10):2044 – 2064, 2010.

Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 22–30. Springer, 2004.

Francisco Herrera, Francisco Charte, Antonio J. Rivera, and María José del Jesús. *Multilabel Classification - Problem Analysis, Metrics and Techniques*. Springer, 2016.

John D. Kelleher, Brian Mac Namee, and Aoife D'Arcy. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. The MIT Press, 2015. ISBN 0262029448, 9780262029445.

Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. ISBN 0070428077, 9780070428072.

Arjun Pakrashi, Derek Greene, and Brian Mac Namee. Benchmarking multi-label classification algorithms. In *Proceedings of the 24th Irish Conference on Artificial Intelligence and*

Cognitive Science, AICS 2016, Dublin, Ireland, September 20-21, 2016., pages 149–160, 2016.

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

Eleftherios Spyromitros, Grigorios Tsoumakas, and Ioannis Vlahavas. An empirical study of lazy multilabel classification algorithms. In *Proc. 5th Hellenic Conference on Artificial Intelligence (SETN 2008)*, 2008.

Grigorios Tsoumakas, Anastasios Dimou, Eleftherios Spyromitros, Vasileios Mezaris, Ioannis Kompatsiaris, and Ioannis Vlahavas. Correlation-based pruning of stacked binary relevance models for multi-label learning. In *Proceedings of the 1st international workshop on learning from multi-label data*, pages 101–116, 2009.

Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *J. Mach. Learn. Res.*, 12:2411–2414, 2011.

David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

Z. Younes, F. Abdallah, and T. Denoeux. Multi-label classification algorithm derived from k-nearest neighbor rule with label dependencies. In *2008 16th European Signal Processing Conference*, pages 1–5, Aug 2008.

M. L. Zhang and Z. H. Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, Oct 2006. ISSN 1041-4347.

M. L. Zhang and Z. H. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40:2038–2048, 2007.

M. L. Zhang and Z. H. Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, Aug 2014. ISSN 1041-4347.