Editing training data for multi-label classification with the k-nearest neighbors rule

Sawsan Kanj [a,b], Fahed Abdallah [a], Thierry Denœux [a], Kifah Tout [b]

[a] *Heudiasyc, UMR CNRS 7253, Université de Technologie de Compiègne,*
Rue Roger Couttolenc,
CS 60319,
60203 COMPIEGNE CEDEX,
FRANCE,
Tel : (33) 3 44 23 52 15 fax : (33) 3 44 23 44 77
firstname.lastname@hds.utc.fr

[b] *AZM center for biotechnology research, EDST, Lebanese University,*
Rue miten, face malaab baladi,
Tripoli, Lebanon,
ktout@ul.edu.lb

Abstract

Multi-label classification allows instances to belong to several classes at once. It has received significant attention in machine learning and has found many real world applications in recent years, such as text categorization, automatic video annotation and functional genomics, resulting in the development of many multi-label classification methods. Based on labelled examples in the training dataset, a multi-labelled method extracts inherent information in order to output a function that predicts the labels of unlabelled data. Due to several problems, like errors in the input vectors or in their labels, this information may be wrong and might lead the multi-label algorithm to fail. In this paper, we propose a simple algorithm for overcoming these problems by editing the existing training dataset, and adapting this edited set with different multi-label classification methods. Evaluation on benchmark datasets demonstrates the usefulness and effectiveness of our approach.

Key words: Classification; multi-label; k-nearest neighbors rule; prototype selection; edition.

1 Introduction

Multi-label classification is the supervised classification task where each instance can be associated with multiple classes simultaneously from a set of disjoint classes; the classes are then no longer mutually exclusive. Contrary to single-label classification, the multi-label problem is influenced by intrinsic latent correlations between labels, in the sense that the membership of an instance to a class can be helpful to predict its set of labels [44]. For example, a patient with a high blood pressure is more likely to develop heart disease than an other person, but less likely to develop a muscular dystrophy.

Multi-label classification methods have been applied with modern applications like text categorization, where each document can be associated with a set of predefined topics [30]. In bioinformatics, each protein may be labelled with multiple functional labels such as metabolism, energy and cellular biogenesis [17]. In video annotation, a film might be annotated with several labels or tags [25].

Multi-label methods learn usually a classifier function from the training dataset with known class labels. However, real world data often suffer from noisy or erroneous instances due to several problems, like errors in the input vectors or in their labels. To cope with this problem in the framework of single-label learning, several methods based on data reduction have been introduced. These techniques are usually based on prototype selection [4,13,27,39].

Prototype selection methods are usually applied to remove erroneous or redundant instances from the training dataset [13,20,24]. These methods are widely used with the traditional nearest neighbor rule due to their simplicity and effectiveness. In addition to improving classification accuracy for unseen instances, using prototypes dramatically decreases storage and classification-time costs.

However, despite extensive work in multi-label learning [5,21,26,37,42,44], there is a lack of methods for improving the quality of multi-labelled training instances. This fact motivated us to study this problem in the framework of multi-label learning. In this paper, we develop an original method based on a prototype selection using the nearest neighbor rule and a local evaluation criterion, in order to purify training dataset and improve the performance of multi-label classification algorithms. The evaluation criterion used in this paper is the very well known Hamming loss metric. Nevertheless, the proposed method may be straightforwardly adapted to any other criterion. Given one training instance with known set of labels, we consider the editing rule which, based on the Hamming Loss calculated by estimating to this instance a set of predicted labels from the neighborhood, either delete or remain the instance

unchanged. After applying this edited rule on all observations of the training dataset and eliminating the less relevant in the sense of the chosen criterion, a learning algorithm on the edited training set may be applied efficiently.

To show the effectiveness of this method, we apply existing multi-label classification methods, which are the evidential multi-label k-nearest neighbor (EMLkNN) [10] and the Rank-SVM [12] methods, on the edited dataset. The proposed algorithm is applied to several multi-labelled data from different domains using several multi-label classification measures. Even if the Hamming loss is used as a criterion to edit the data, the performances are evaluated using several multi-label classification measures. Note that, more than increasing classification performance, the new method has the advantage of needing less of storage requirements and decreasing the running time of the initial classification algorithms.

Note that a short paper on the purification (or edition) of multi-labelled datasets was presented at the conference Fusion [18]. In this paper, the edition method is more thoroughly interpreted and discussed. Furthermore, we add the effect of editing on the SVM techniques and we provide an illustrative example on a simulated dataset. In addition, extensive comparisons on several real world datasets are presented, and the effectiveness of the method compared to that before editing is shown using statistical tests (t-test and Friedman test).

This paper is structured as follows. Background notions on the nearest neighbor rule in the classical single-label framework and some related techniques for prototype selection will first be recalled in Section 2. Section 3 will introduce the principle of multi-label classification and review the EMLkNN and Rank-SVM methods. Our approach will then be exposed in Section 4. Section 5 will report the experimental evaluation of the presented methods on synthetic and real-world datasets. Finally, our contribution will be summarized in Section 6.

2 Related work on prototype selection for single-labelled data

The problem of noise handling has received considerable attention in the literature on machine learning. Seeking to start with something relatively simple, scientists have focused on the nearest neighbor classifier considered as one of the most well-known technique in machine learning and data mining due to its simplicity and effectiveness. Given a training set of single-labelled data, the idea is to select an optimal set of training instances, known as prototypes, in order to maximize the performances of the Nearest Neighbor (NN) classifier and/or to minimize the computing time of this classifier [8]. Later, the idea of selecting "good" instances has also been applied to other types of classi-

fiers [4]. In this section, we will rapidly review the nearest neighbor rule, and give a definition and summary of work related to prototype selection methods for the NN rule.

2.1 Nearest Neighbor classification

The Nearest Neighbor rule [7] is a well-known and non-parametric decision procedure for machine learning and data mining tasks. It has been considered as one of the most effective algorithms in machine learning, and one of the top ten methods in data mining [13,41]. In traditional supervised learning, this rule assigns to an unseen sample $\mathbf{x}$, the class of the nearest training instance according to some distance metric. The voting k-nearest neighbor rule (k-NN), with $k > 1$, is a generalization of the NN approach where the predicted class of $\mathbf{x}$ is set as equal to the class represented a majority of its k nearest neighbors in the training set.

However, the k-NN rule suffers from several problems such as large storage requirements, high computational complexity in the operational phase, and low tolerance to noise due to considering all instances as relevant while the training set may contain noisy or mislabelled examples. Different techniques have been proposed in the literature to alleviate these problems. One technique, known as prototype selection, consists of selecting an appropriate subset of the training data that yields a similar or even higher classification accuracy. Prototype selection methods can be categorized into three different families. First, *edition methods* eliminate noisy instances from the original training set in order to improve classification accuracy. Second, *condensation methods* select a sufficiently small subset of training instances which lead to the same performance of the single nearest neighbor rule (1-NN), by removing instances that will not affect classification accuracy. Finally, *hybrid methods* select a small subset of training instances that incorporates the goals of these two previous methods [13,3]. In the following, we consider only the editing methods.

2.2 Editing methods

Editing methods process the training data by removing border and noisy instances or making other necessary cleaning, with the aim of improving classification accuracy of learning algorithms on test data. Below we review some algorithms related to the editing approach for the nearest neighbor rule.

Wilson proposed the first editing rule [40], called Edited Nearest Neighbor (ENN), to improve the performance of the 1-NN rule. This method can be

described in the following manner. Each instance in the training set is classified using the k-NN rule, and it is marked for deletion if its predicted class does not agree with the true class. Edition is achieved by deleting all misclassified instances at once. After, any input sample is classified using the 1-NN rule with the remaining instances. Experiments with the editing rule were reported by Tomek who proposed two variants of the ENN rule: RENN and All k-NN [34]. The Repeated Edited Nearest Neighbor (RENN) rule repeats the ENN algorithm until a stable set is obtained where no more samples are edited out. The All k-NN applies iteratively the ENN algorithm with the i-NN rule where i is going from 1 to k.

In [19], the generalized editing procedure based on the kk'-NN rule was introduced. The purpose of this procedure was two-fold: improving the level of performance of the ENN algorithm and reducing the proportion of deleted samples. Based on the class of a majority of k' instances from a group of k nearest samples to an instance $\mathbf{x}$, the group of k samples is either deleted or relabelled as belonging to the majority class. The 1-NN is then used on the edited set to classify an input instance. In [11], the authors proposed the well-known Multiedit algorithm, which randomly breaks the initial training set into different subsets. In each subset, every instance is classified using the 1-NN rule with the instances in the next subset. Misclassified instances are discarded. The remaining instances constitute a new set and the algorithm is iteratively repeated until no more instances are edited out.

In [16], a Modified Edited k-NN rule (MEKNN) was proposed. According to this rule, a sample $\mathbf{x}$ is deleted from the initial set if its class does agree with the class of its k nearest neighbors and their tying instances (tying instances are those in the training set that are at the same distance to $\mathbf{x}$ as its furthest neighbor). In addition, this method introduces a fixed number of pairs (k, k'). k is the number of neighbors to make the edition process and k' is employed to classify any new instance in the obtained edited set. The goal was to obtain the optimal pairs of k and k' to employ the final editing reference set.

Another method for nearest neighbor editing was proposed in [15]. This method uses the concept of semi-supervised learning and edits the training instances by using the whole dataset including: labelled and unlabelled instances. The proposed method, called NNEAUD (Nearest Neighbor Editing Aided by Unlabelled Data), consists of two steps: labels are first predicted for unlabelled instances, and the augmented dataset is then used in data editing. The NNEAUD method uses ENN, RENN, and All$k$NN algorithms with unlabelled data to edit the training instances.

3 Multi-label learning

3.1 Problem

Let $\mathbb{X}$ denote an instance space, and let $\mathcal{Y} = \{\omega_1, \ldots, \omega_Q\}$ be a finite set of labels. Let $\mathcal{D} = \{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_n, Y_n)\}$ denote a dataset composed of n multi-labeled objects $(\mathbf{x}_i, Y_i)$, $\mathbf{x}_i \in \mathbb{X}$ and $Y_i \subseteq \mathcal{Y}$, where each instance is independent and identically distributed (i.i.d.) drawn from an unknown distribution. The goal of multi-label learning is to build a multi-label classifier $\mathcal{H}$ that maps an instance $\mathbf{x}$ to its associated set of labels Y and optimizes some evaluation metrics. Here, the set of all subsets of $\mathcal{Y}$ is the power set of $\mathcal{Y}$ denoted by $2^{\mathcal{Y}}$.

Numerous methods have been proposed in the literature to deal with multi-label learning problems. Existing algorithms can be grouped into three categories as proposed in [21]: problem transformation approaches, problem adaptation algorithms and ensemble methods. The first category divides the multi-label problem into one or more conventional single-label problems. Binary Relevance and Label Powerset are two examples of such type of approaches. The second category generalizes single-label algorithms to cope with multi-labeled data directly. Examples include boosting [29], decision tree [2] and the Multi-label k-nearest neighbors methods [44,47]. Finally, the third category incorporates the merits of these two previous approaches. Several ensemble methods have been proposed, among them: ensemble of classifier chains [26], random k-label sets [38] and ensemble of multi-label classifiers [32].

3.2 Performance evaluation in multi-label learning

In the traditional single-label classification task, predictive performance is determined under the traditional *accuracy* measure, where each test instance can either be correctly or incorrectly classified, and performance is given by the proportion of correctly classified test instances. In the multi-label classification task, predictive performance is more complex than that of single-label systems, where the classification of each test instance can be fully correct, partially correct or fully wrong. Given a set $\mathcal{S} = \{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_m, Y_m)\}$ of m test examples, evaluation metrics can be divided into two groups: *prediction-based* and *ranking-based* metrics [37]. *Prediction-based* metrics are calculated based on the comparison between the predicted and the ground truth sets of labels, while *ranking-based* metrics evaluate the label ranking quality depending on a scoring function $f(.,.)$, ($f : \mathbb{X} \times \mathcal{Y} \longrightarrow \mathbb{R}$, where $\mathbb{X}$ is the domain of instances and $\mathcal{Y}$ is the set of Q target classes) that attributes a score to each class in $\mathcal{Y}$ [44]. More details on evaluation metrics are given in appendix A.

We will focus in this paper on the Hamming loss and the Ranking loss metrics. The Hamming loss is a prediction-based metric regarded as an average of the error rate of the classifier on the Q binary problems where the decision is performed separately [28]. It is defined by:

$$\mathcal{H}Loss = \frac{1}{m}\sum_{i=1}^{m}\frac{|Y_i \triangle \widehat{Y}_i|}{Q}, \tag{1}$$

where Y_i is the ground truth label set for the pattern $\mathbf{x}_i$, $\widehat{Y}_i$ is the predicted label set for $\mathbf{x}_i$ and $\triangle$ denotes the symmetric difference between two sets. In other words, the Hamming loss is based on counting prediction errors (an incorrect label is predicted) and missing errors (a true label is not predicted). The Ranking loss is a ranking-based metric which evaluates the average fraction of crucial pairs of labels that are misordered for an instance [29]. The *Ranking Loss* is:

$$\mathcal{R}Loss = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|Y_i|\overline{Y_i}|}|R(\mathbf{x}_i)|, \tag{2}$$

where $R(\mathbf{x}_i) = \{(\omega_q, \omega_r) \in Y_i \times \overline{Y_i} \mid f(\mathbf{x}_i, \omega_q) \leq f(\mathbf{x}_i, \omega_r)\}$, $\overline{Y_i}$ denotes the complement of Y_i in $\mathcal{Y}$. Smaller values of these metrics correspond to higher classification quality. Note that the value of these evaluation criteria is in the interval [0, 1]. We will present briefly in next sections the two multi-label algorithms that will be used in this paper and which are the Evidential multi-label k-NN and the Rank-SVM methods.

3.3 Evidential multi-label k-NN classification

The evidential k-NN (EMLkNN) method introduced in [10] answers the multi-label classification problems under the belief functions framework and can be summarized as follows. Let $\mathcal{D} = \{(\mathbf{x}_1, A_1, B_1), \ldots, (\mathbf{x}_n, A_n, B_n)\}$ be the learning set, where $A_i \subseteq \mathcal{Y} = \{\omega_1, \ldots, \omega_Q\}$ denotes a set of classes that surely apply to the instance $\mathbf{x}_i$, and B_i is the complement of A_i in $\mathcal{Y}$, ($\mathcal{Y}$ is known as the *frame of discernment* of the problem).

To classify an unlabelled instance $\mathbf{x}$, we identify its k-nearest neighbors, denoted as $\mathcal{N}_{\mathbf{x}}$, by computing the distance of $\mathbf{x}$ to the labelled objects in $\mathcal{D}$ based on a certain distance function. Each element $\mathbf{x}_i$ in $\mathcal{N}_{\mathbf{x}}$ constitutes an item of evidence regarding the label set of $\mathbf{x}$. This item of evidence can be described by the following simple two-valued mass function:

$$\begin{aligned} m_i(A_i, B_i) &= \alpha \exp(-\gamma d(\mathbf{x}, \mathbf{x}_i)), \\ m_i(\emptyset, \emptyset) &= 1 - \alpha \exp(-\gamma d(\mathbf{x}, \mathbf{x}_i)), \end{aligned} \tag{3}$$

where $d(\mathbf{x}, \mathbf{x}_i)$ is the distance between $\mathbf{x}$ and $\mathbf{x}_i$, α and γ are two parameters, such that $0 < \alpha < 1$ and $\gamma > 0$. Parameter α is usually fixed to a value close to 1 such as 0.95 [9], whereas γ can be optimized or fixed heuristically [48]. If the number of neighbors of the $\mathbf{x}$ is k, the resulting k mass functions are combined using the conjunctive rule:

$$m = \textcircled{\cap}_{i:\ \mathbf{x}_i \in \mathcal{N}_\mathbf{x}} m_i \tag{4}$$

where the $\textcircled{\cap}$ symbol denotes the unnormalized Dempster's rule of combination [10]. This rule strongly emphasizes the agreement between multiple sources, where no elementary item of evidence should be counted twice. The predicted multi-label set for $\mathbf{x}$ is then determined by computing separately for each label $\omega \in \Omega$ two quantities: the degree of belief $bel(\{\omega\}, \emptyset)$ that the true label set Y contains ω, and the degree of belief $bel(\emptyset, \{\omega\})$ that it does not contain ω. The multi-label classifier $\mathcal{H}$ is defined finally as:

$$\mathcal{H}(\mathbf{x}) = \{\omega \in \mathcal{Y} | bel(\{\omega\}, \emptyset) \geq bel(\emptyset, \{\omega\})\}, \tag{5}$$

where $\emptyset$ denotes the empty set of $\mathcal{Y}$.

3.4 Rank-SVM

Rank-SVM is a multi-label ranking approach introduced by Elisseeff and Weston in [12]. The ultimate goal was to minimize a criterion measure for multi-label learning, called Ranking loss, and to maximize the margin. The authors introduce a special multi-label margin defined on $(\mathbf{x}, Y)$ as the signed distance between the instance $\mathbf{x}$ and the decision boundary. Note that the boundary of class q is a grouping of several boundaries separating the class q and the other classes. For Rank SVM method, which ranks the values of $r_q(x) = \langle w_q, \mathbf{x} \rangle + b_q$, the decision boundaries for $\mathbf{x}$ are defined by the hyperplanes whose equations are $\langle w_q - w_l, \mathbf{x} \rangle + b_q - b_l = 0$. Thus, the margin with respect to class q is equal to:

$$\min_{(q,l)\ |\ (\omega_q, \omega_l) \in (Y \times \overline{Y})} y_q \frac{\langle w_q - w_l, \mathbf{x} \rangle + b_q - b_l}{\|w_q - w_l\|}$$

where w_q, w_l and b_q, b_l denote the weight vectors and bias terms, and y_q is a binary element equal to $+1$ if label q is in Y, -1 otherwise. According to [12], q denotes a relevant label, and l the irrelevant one. For training instances, it is desirable that any relevant label should be ranked higher than any irrelevant one.

The Rank-SVM model is built from two different sub-systems. The first one, named ranking system, orders the labels via a quadratic optimization problem, according to their outputs, $r_q(x) = \langle w_q, \mathbf{x} \rangle + b_q$ for $q = 1, \ldots, Q$. The other goal of this method is to predict a threshold $t(\mathbf{x})$ and all integer q such that

$r_q(\mathbf{x}) > t(\mathbf{x})$ are considered to belong to the label set Y of $\mathbf{x}$. It is well-known that such an algorithm can be generalized to non-linear separating boundaries by just replacing the dot products $< \mathbf{x}_i, \mathbf{x}_j >$ by kernels $k(\mathbf{x}_i, \mathbf{x}_j)$.

4 Editing multi-labelled data using the k-NN rule

4.1 Motivation

In multi-label learning, the goal is to generate a multi-label classifier that will generalize from a set of multi-labelled training instances in such a way that classification performances for labelling new data are optimized. However, errors in multi-labelled training datasets can occur for several reasons. One cause is the subjectivity, when the boundaries of each class are based on individual perspectives. For example, in genre classification of musical signals, each musical genre may have its boundaries shifted from person to person [1]. A second cause of anomalies or noisy instances is ambiguity during data-entry. For example, in clinical text for multi-label classification (medical multi-labelled data collected from Cincinnati children's hospital medical center), abbreviations and acronyms used to anonymization of patients may lead to ambiguity when processing such data by taking more than one sense and having multi-purposes (in a clinical setting, *FT* can be an abbreviation for *full-term*, *foot test*, *field test*, *full-time* or *family therapy*) [23]. Other errors can arise from missing information and data transformation or storage. Furthermore, many examples may have an erroneous set of labels due to an experimental assignment problem or even a human annotation error. To the best of our knowledge, no algorithm addressing these problems under the multi-label framework has been proposed so far.

In the following, we propose an original method to edit multi-labelled data by identifying and eliminating erroneous or anomalous samples. The purpose of this method is three-fold: first, to increase the quality of training instances assumed to become more reliable; second, to improve the performances of the classifier built from the resulting training data; and third to increase the response time of the learning algorithm. This method is based on the k-nearest neighbor rule for multi-label classification, and on an evaluation criterion used locally in the set $\mathcal{N}_{\mathbf{x}}$ of k-nearest neighbors of $\mathbf{x}$ to evaluate the quality of an instance $\mathbf{x}$. Based on this evaluation criterion, we can delete the most irrelevant, or the *worst* samples from the initial training dataset. We will present hereafter a simple method using this metric conjointly with a k-NN rule for multi-label classification in order to edit the training dataset.

4.2 Editing algorithm: Edited Nearest Neighbor for Multi-Labelled data

Let $\mathbf{x}$ be an unseen instance for which we wish to estimate the set of labels. In the following steps, we describe the proposed method to edit the training dataset:

(1) For each training instance $\mathbf{x}_i$ in $\mathcal{D}$, search for $\mathcal{N}_{\mathbf{x}_i}$, the set of its k nearest neighbors;
(2) Apply a k-NN based multi-label classifier and calculate a predicted set of labels $\widehat{Y}_i$ for $\mathbf{x}_i$;
(3) For each training instance in $\mathcal{D}$, calculate the associated Hamming loss given by:

$$\mathcal{H}Loss_i = \frac{|Y_i \triangle \widehat{Y}_i|}{Q}; \tag{6}$$

(4) Estimate the Hamming loss $HLoss$, which is the mean of the associated Hamming loss for all instances in $\mathcal{D}$:
- if $HLoss$ is less than a predefined threshold t, then stop the algorithm;
- else,
  (a) Rank the training instances in $\mathcal{D}$ with respect to their $HLoss_i$ and select a subset $\mathcal{E}^l$ containing l instances with the higher Hamming loss $HLoss_i$;
  (b) Update the training set by deleting those in $\mathcal{E}^l : \mathcal{D} \leftarrow \mathcal{D} \setminus \mathcal{E}^l$;
  (c) Return to step 1.

Note that any k-NN based multi-label classifier [44,10,46] can be applied in Step 2. In this paper, we chose the EMLkNN method introduced in Section 3.3. According to this method, each element in $\mathcal{N}_{\mathbf{x}_i}$ represents a piece of knowledge about the labelling of $\mathbf{x}_i$. A two-valued mass function is then associated to each of the k neighbors in $\mathcal{N}_{\mathbf{x}_i}$ according to Equation 3. These items of evidence are combined to produce a global mass function, using the conjunctive rule of Equation 4. In order to estimate the label set for $\mathbf{x}_i$ denoted by $\widehat{Y}_i$, the global mass function is used according to Equation 5. Intuitively, k should be set to a small value because if k is high, undesirable instances elimination will occur on the boundary between different classes. If k is equal to 1, the EMLkNN algorithm boils down to the 1-NN algorithm, and the set of labels to be assigned to an example is the same as that of his neighbor. In Step 3, one can use other stopping criteria than the general $HLoss$. For example, we can stop editing if the Hamming loss associated to each instance is less than a predefined threshold t. We can also substitute the Hamming loss by another multi-label metric evaluation. In Steps 4a and 4b, we delete instances with high value of $HLoss_i$, which means deleting the *worst* instances with respect to a local EMLkNN rule. One can add a condition to keep instances belonging to classes with low occurrence.

5 Experimental Evaluation

In this section, we present experiment results with synthetic and real-world datasets from different domains to demonstrate the effect of edition on the performances of the two multi-label classification methods described below.

5.1 Experiments with Synthetic Data

In this section, we will illustrate the behavior of our editing algorithm on synthetic datasets using the two methods of classification discussed above. The goal of these experiments is to study the effects of edition on multi-label learning algorithms.

A dataset with three-overlapping classes in two-dimension was first considered. The dataset contains 600 instances belonging to three possible labels $\Omega = \{\omega_1, \omega_2, \omega_3\}$. These instances were drawn from seven Gaussian distributions with means $(-5, -5)$, $(5, -5)$, $(0, 5)$, $(0, -5)$, $(-3, 1)$,$(3, 1)$, and $(0, 0)$. The standard deviations was equal two for the first three distributions and one for the others. We assigned the following classes, respectively, for samples drawn from each of these distributions: $\{\omega_1\}$, $\{\omega_2\}$, $\{\omega_3\}$, $\{\omega_1, \omega_2\}$, $\{\omega_1, \omega_3\}$, $\{\omega_2, \omega_3\}$, $\{\omega_1, \omega_2, \omega_3\}$. This dataset was randomly divided into training and test datasets with size 400 and 200, respectively. Table 1 gives the distribution of instances over the different labels.

Table 1
Description of the synthetic data without the erroneous instances.

Label set	Training instances	Testing instances
$\{\omega_1\}$	85	41
$\{\omega_2\}$	84	41
$\{\omega_3\}$	82	46
$\{\omega_1, \omega_2\}$	30	20
$\{\omega_1, \omega_3\}$	42	18
$\{\omega_2, \omega_3\}$	46	23
$\{\omega_1, \omega_2, \omega_3\}$	31	11

To test our editing algorithm, 40 instances drawn in the region allocated to classes $\{\omega_1\}, \{\omega_2\}$ and $\{\omega_1, \omega_2\}$ were wrongly assigned to class $\{\omega_3\}$. These noisy samples are generated randomly from two normal distributions with means $(-4, -6)$ and $(4, -6)$, respectively, and a standard deviation equal to 2. Figure 1 shows the dataset (initial + noisy instances) with their class as-

signments.

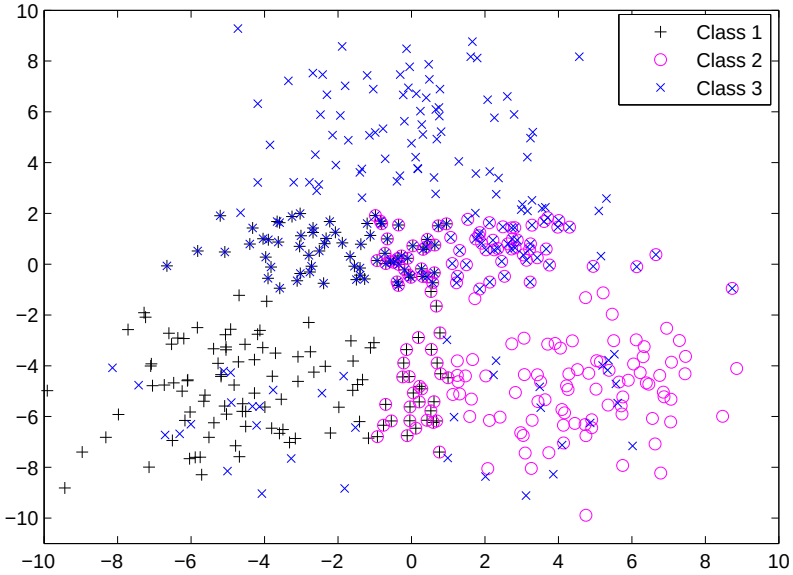
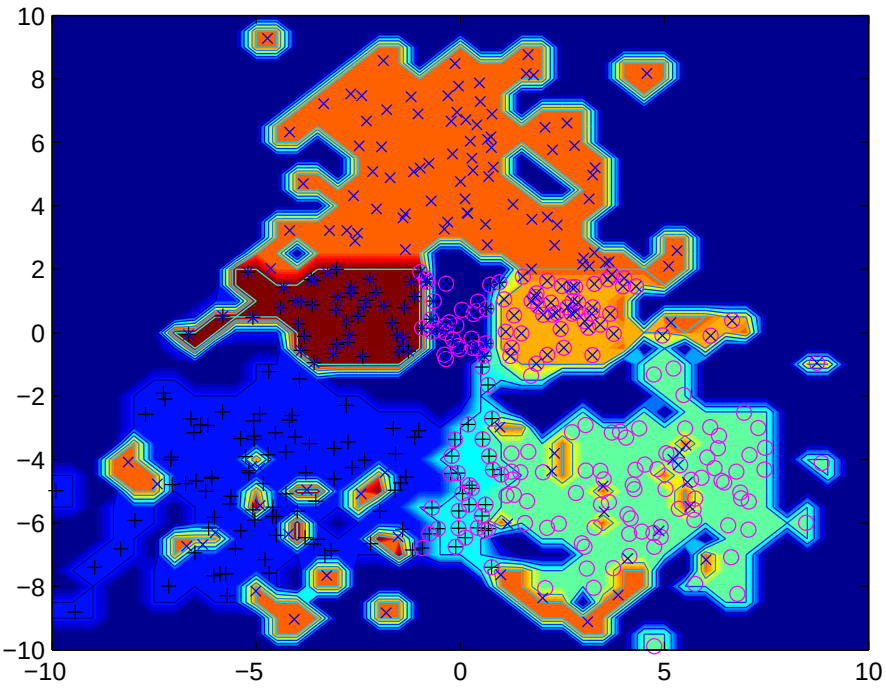


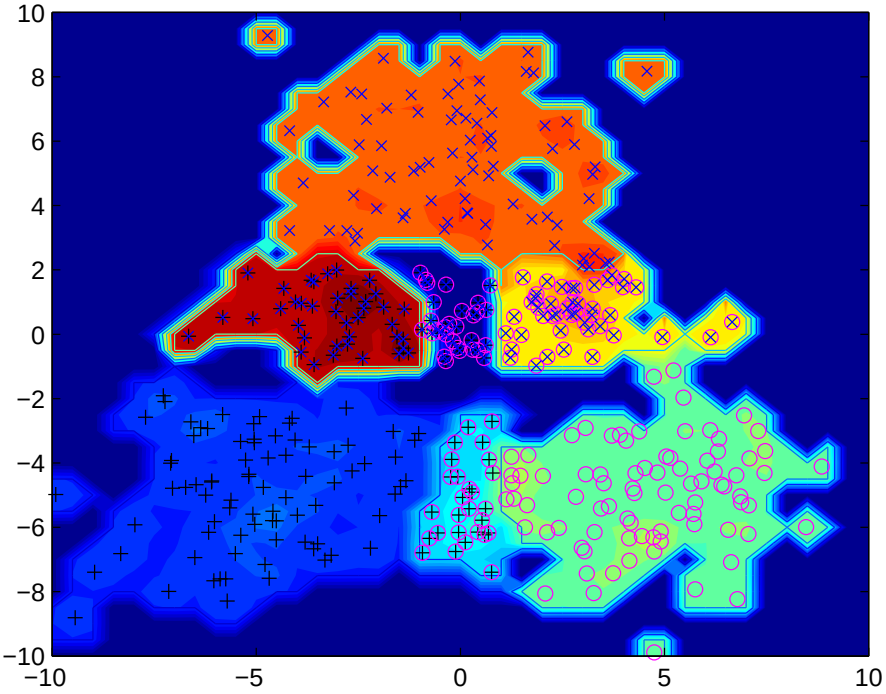
Fig. 1. Training instances of synthetic data

Figures 2(a) and 2(b) show the decision boundaries for our synthetic data with a support vector domain using a Gaussian kernel. The boundary region for each class label was drawn using the Rank-SVM method, with the Gaussian kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma_r \|\mathbf{x} - \mathbf{x}'\|^2)$, $\gamma_r = 5$. We used the same parameters values for the Rank-SVM method with the training data before and after editing. Figure 2(a) shows the decision boundaries for the initial training dataset. As we can see, these decision boundaries are significantly influenced by noisy instances and there is no clear separation between classes. In the area of class $\{\omega_2\}$ (on the right of this figure), we can see several zones belonging to classes $\{\omega_1, \omega_2, \omega_3\}$. Also, in the area of class $\{\omega_1\}$, the erroneous instances create many zones with instances belonging to class $\{\omega_3\}$.

In Figure 2(b), we can see the decision boundaries for the same dataset after editing. In this figure, we can see that, with editing, noisy instances have been removed, and, smoother decision boundaries are produced. The area is now divided into seven zones. Instances belonging to class $\{\omega_1\}$ are on the left of Figure 2(b), instances assigned by label $\{\omega_2\}$ are on the right, and instances labelled with $\{\omega_3\}$ are at the top. Using a geometrical interpretation, we can easily distinguish the area belonging to each combination of these classes. Instances annotated by three classes are in the middle of this figure. Note that the number of training instances was reduced to 382 (initial number of training data was 440), and the number of support vectors was decreased from 409 to 352. Table 2 reports the experimental results on three evaluation



(a)



(b)

Fig. 2. The Rank-SVM decision boundaries between classes with the training instances, a) before editing, b) after editing.

criteria: Hamming loss, accuracy and the F1-measure.

Table 2
Some evaluation measures for the Rank-SVM method before and after the edition of the synthetic dataset.

	Before Editing	After Editing
Hamming loss^{-}	0.1667	**0.1017**
Accuracy^{+}	0.7283	**0.8358**
F1^{+}	0.7422	**0.8423**

+(-): the higher (smaller) the value, the better the performance.

From these two figures, we can observe that training the Rank-SVM method with a purified dataset leads to smoother separating boundaries, creates homogeneous clusters and reduces the number of support vectors.

Figure 3 shows the performance of our editing approach on the synthetic data using the EMLkNN method. We used from the library of multi-label measures three evaluation criteria: Hamming loss, accuracy and the F1-measure. The values of these metrics are shown as a function of the number of neighbors k. From this figure, we can observe that when k takes small values, the EMLkNN algorithm tested on the edited dataset performs better than EMLkNN tested on noisy dataset. As k increases, the EMLkNN method tends to have the same performance on these two datasets . This can be explained by the fact that, when increasing the number of neighbors, the effect of randomly erroneous instances decreases giving that we use more information (coming from more instances), and also the applied method (EMLkNN) is based on an evidential distance-weighted k-nearest neighbor rule.

5.2 Experiments on Real-World Data

In this section, we apply the two multi-label classification methods discussed above (EMLkNN and Rank-SVM) to our datasets and we evaluate their performances before and after editing. In the following, we will report the benchmark datasets, the evaluation metrics used in our experiments and parameter settings for edition. Finally, we will provide a discussion of experimental results.

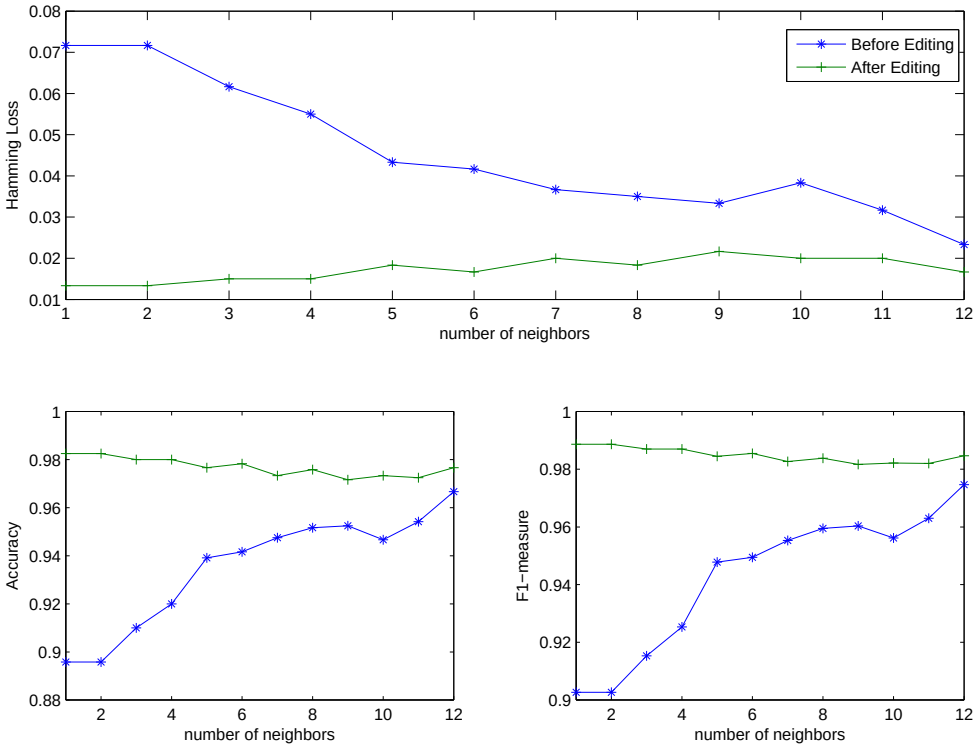


Fig. 3. Some evaluation measures for the EMLkNN method before and after the edition of the synthetic dataset.

5.2.1 Datasets

The datasets[1] that were included in our experiments cover different application domains: multimedia classification (Emotions), bioinformatics (Yeast) and text categorization (Medical, Enron and Webpage).

- *Emotions dataset.* This dataset consists of 593 songs labeled by experts according to the emotions they generated. Each piece of music is described by 8 rhythmic features and 64 timbre features, and can be annotated with the following emotions: *amazed-surprised, happy-pleased, relaxed-calm, quiet-still, sad-lonely and angry-fearful.* The average number of labels for each song is 1.869, and the number of distinct label sets is equal to 27 [35].
- *Yeast dataset.* The *yeast Saccharomyces cerevisiae* is one of the best studied organisms. Each gene is described by the concatenation of micro-array expression data and phylogenetic profile and it is associated with a subset of 14 functional classes from the Comprehensive Yeast Genome Database

[1] Datasets available at `http://mulan.sourceforge.net/datasets.html`, and `http://cse.seu.edu.cn/people/zhangml/`.

of the Munich Information Center for Protein Sequences [2]. This dataset contains 2417 genes and 14 possible labels [22].

- *Medical dataset.* This dataset contains 978 documents for patient symptom histories collected from the *Computational Medicine Center* concerning a challenge task on the automated processing of clinical free text. Each document is represented by a vector of 1449 features [23].
- *Enron dataset.* The *Enron email* [3] dataset was made public by the Federal Energy Regulatory Commission during its investigation. It contains around 517.431 emails (without attachments) from 151 users distributed in 3500 folders. Each message includes the senders and the receiver email address, date and time, subject, body, text and some other email specific technical details. After preprocessing and careful selection of these documents, 53 different labels are obtained with 753 combinations of distinct label sets [31].
- *Webpage categorization dataset.* This dataset were collected from the "yahoo.com" domain [33]. Eleven different webpage categorization subproblems are considered, corresponding to 11 independent multi-label categories: *Arts and Humanities, Business and Economy, Computers and Internet, Education, Entertainment, Health, Recreation and Sports, Reference, Science, Social and Science, and Society and Culture*. Each subproblem consists of 5000 documents (2000 as training dataset and 3000 as testing dataset). Each webpage was represented as a bag of words and normalized to the unit length.

Tables 3 and 4 provide an overview of the different characteristics of all experimental datasets. These characteristics are explained in the appendix B at the end of the article.

Table 3
Characteristics of the Emotions, Yeast, Medical and Enron datasets.

	Domain	Number of instances	Feature vector dimension	Number of labels	Label cardinality	Label density	Distinct label sets
Emotions	music	593	72	6	1.868	0.311	27
Yeast	biology	2417	103	14	4.237	0.303	198
Medical	text	978	1449	45	1.245	0.028	94
Enron	text	1702	1001	53	3.378	0.064	753

5.3 Parameter Tuning

In this section, we comment how to tune different parameters to apply the different algorithms described in this paper. We call *editing* parameters those applied on the initial dataset with the editing algorithm in order to obtain an edited training dataset. We call *testing* parameters those used in a multi-label

[2] `http://mips.gsf.de/genre/proj/yeast/`
[3] `http://enrondata.org/content/research/`

Table 4
Characteristics of the Webpage categorization dataset.

	Number of instances	Feature vector dimension	Number of labels	Label cardinality	Label density	Distinct label sets
Arts and Humanities	5000	462	26	1.627	0.063	462
Business and Economy	5000	438	30	1.590	0.053	161
Computers and Internet	5000	681	33	1.487	0.046	253
Education	5000	550	33	1.465	0.044	308
Entertainment	5000	640	21	1.426	0.068	232
Health	5000	612	32	1.667	0.052	257
Recreation and Sports	5000	606	22	1.414	0.065	322
Reference	5000	793	33	1.159	0.035	217
Science	5000	743	40	1.489	0.036	398
Social and Science	5000	1047	39	1.274	0.033	226
Society and Culture	5000	636	27	1.705	0.063	582

classification algorithm learnt from initial or edited learning dataset. Note that the number k of neighbors to be used is not necessarily the same as that used in the editing algorithm. To avoid confusion, the number of neighbors used in the editing algorithm will be noted by k'. Hereafter, we will show the influence of these parameters by using the *Emotions* dataset.

5.3.1 Editing parameters

For the editing algorithm presented in Section 4.2, there are three tunable parameters:

- γ: Parameter used in Equation 3 to scale the distance to each neighbor. It was fixed at the best value obtained by cross validation using the EMLkNN method on the initial training dataset.
- k': Number of neighbors used in the editing algorithm.
- t: Threshold used to determine the number l of instances to delete. We use in the simulation a Hamming loss calculated on each instance as in Equation (6). This Hamming loss calculated on only one instance will have a value equals q/Q, where $q \in \{0, \ldots, Q\}$. Note that the value of the parameter t to be taken should depend on the global Hamming loss calculated on the training dataset.

Figure 4 shows the box plot for the Hamming loss metric obtained by the EMLkNN method on the initial dataset before editing the data for different values of γ, where k was varied from 1 to 12 (thus each box plot corresponds to 12 values of the Hamming loss obtained for a given γ). Figure **??** shows the Hamming loss measure obtained as a function of t, where k' was varied from 1 to 12, γ was fixed to 0.1, and k was fixed to 3 (we can get the same results for any value of k, for that we chose for it a small value). The box plot in Figure

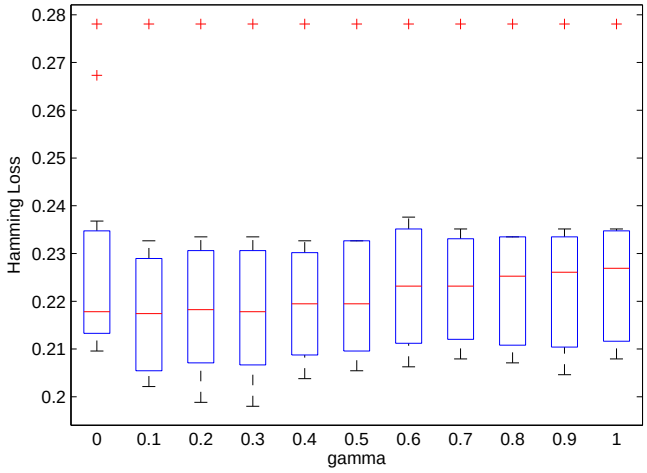


Fig. 4. Hamming loss measure for EMLkNN on the initial Emotions training set for different values of γ.

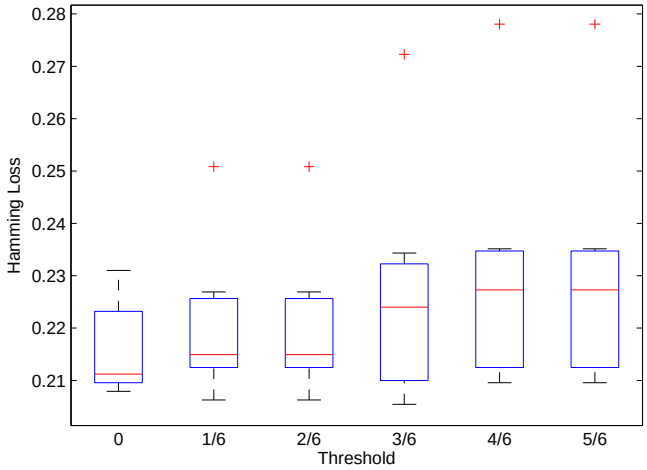


Fig. 5. Hamming loss measure for EMLkNN after editing the Emotions training set as a function of t.

6 shows the Hamming loss criterion with respect to the number of neighbors k'. k was varied from 1 to 12, and γ was fixed to 0.1.

5.3.2 Testing parameters

In the testing phase, the EMLkNN and the Rank-SVM methods are tested with the edited data. EMLkNN has two parameters: the number of neighbors k, and the discounting parameter γ. These parameters were determined using grid search and by focusing on the Hamming loss measure: k was varied from

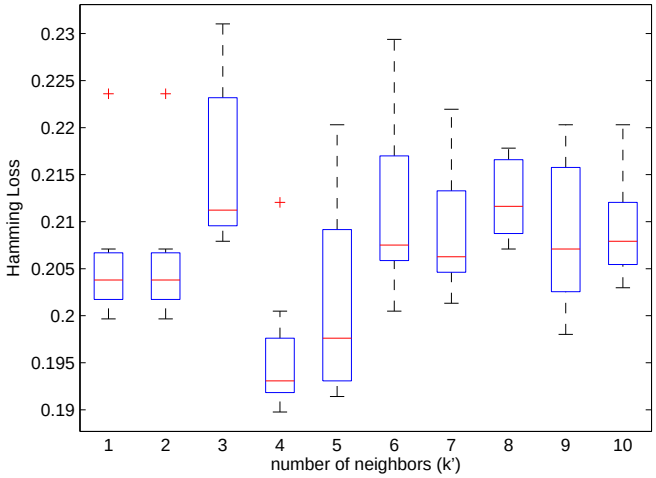


Fig. 6. Hamming loss measure for EMLkNN on the edited Emotions training set as a function of k'.

1 to 12, and γ from 0 to 1 with 0.01 steps. Note that the algorithm presented in Section 4.2 was repeated only once by taking the best value of t; i.e., the one that eliminates an important number of erroneous instances at once.

For the Rank-SVM method, we used the Gaussian kernel with three tunable parameters: kernel scale parameter γ_r, penalty constant C, and maximal iterations M. By focusing on the Hamming loss measure, cross-validation via grid search was applied for parameter tuning as explained in [43]. The γ_R and C parameters took values from $2^7, 2^6, \ldots,$ to 2^{-7} respectively. M was set to 50, 100, 150 and 200.

5.4 Results and Discussion

In this section, we evaluate the performance of the editing algorithm by comparing the results achieved by the EMLkNN and Rank-SVM methods before and after editing. Using the optimal parameter values obtained on the training datasets, we studied the performance using independent test datasets. The experimental results on datasets are given in Tables 5-8. For the webpage dataset, the average performance out of the 11 different categorization problems is reported in Table 9. The rank of each method is given, and the best value on each evaluation criterion is highlighted in bold letters. These results can be summarized as follows:

- From the results in Tables 5-9, we can observe that EMLkNN applied on edited data improves the performance of the same method on the initial dataset for all prediction-based metrics except the Hamming loss measure.

The Hamming loss criterion is similar before and after edition.

- Regarding the Rank-SVM method, results on editing datasets are better than those on initial datasets for all measures (prediction-based and ranking-based metrics).
- The Rank-SVM applied on editing datasets gives the best performance according to the majority of evaluation measures for the Yeast, Medical, Enron,and Webpage datasets. For the Emotions dataset, the best performance on the ranking-based measures was obtained by the Rank-SVM method applied to the edited dataset, while the best results according to predicted-based measures were obtained by the EMLkNN algorithm applied to the edited dataset.

In order to show the effect of edition on data storage, Table 10 reports information about the used datasets: the number of instances of full and edited training datasets, and the time necessary to editing each of these datasets. The results indicate that the edited datasets require less storage space than do the initial datasets.

In order to study the impact of edition on classification time, we compared the time used by each method (programmed in Matlab) applied to the initial and edited datasets. Table 11 presents the total running time (learning + testing time) using the initial training and edited datasets. We can see that the running time of the two classifiers (EMLkNN and Rank-SVM) are significantly reduced in our experiments, except for the Enron dataset. In general, EMLkNN is faster than Rank-SVM, due to the space complexity of the Rank-SVM method which is proportional to $n * Q^2$. The machine used was Intel(R) Xeon(R) CPU at 2.67 GHz, 12 GB RAM with Matlab2012a.

To statistically measure the significance of performance difference between results on initial datasets and those on edited datasets, two statistical indicators are carried out using ten-fold cross validation: the *pairwise t-tests* and the *Friedman test* [14]. The average results of different evaluation criteria using ten-fold cross validation are reported in Tables 12-16 in which we use the pairwise *t-test*. The significance is usually determined under a significance level of $\alpha = 0.05$. To be able to see *the power* of our conclusions, the *p-values* of *Friedman test* and *t-test* on the different datasets are indicated in Tables 17-21. Note that small values for the *p-values* indicate strong presumption against null hypothesis, which is the hypothesis saying that the results of the two methods are not different. For the chosen significance level, we can consider that the results of the two methods (before and after editing) are different if *p-value* < 0.05. We can see also from the tables that the results of the *pairwise t-test* and the *Friedman test* are correlated leading almost to similar conclusion about the significance of the difference between methods.

The results presented in this section show the advantage of editing multi-label datasets to improve the performance of multi-label classifiers. By comparing the performance of multi-label classifiers (EMLkNN and Rank-SVM) before and after edition, we can conclude that editing initial multi-label datasets improve the performance evaluation of some classifiers. Furthermore, we may reduce the complexity of classifiers since we need to train less instances, which are distributed into more homogeneous clusters. We might deduce also that editing training datasets is a way to reduce the running time complexity of some multi-label classification methods. Even if we use the Hamming loss criterion to edit the training datasets, we can get better performance on other metrics.

Note that we tested the use of the edited data set on other multi-label classifiers, namely the C4.5 based on decision trees [6], and the $MLMLP$ based on neural network [45]. The results we obtained show better performances of these methods on all the datasets, which follows the behavior of the previous presented methods (EMLkNN and Rank-SVM). We are not showing the numerical results for a better readability of the paper.

6 Conclusion

In this paper, we have addressed the problem of prototype selection in the framework of multi-label learning. Although the extensive work in multi-label classification, to the best of our knowledge, the topic of prototype selection has not received any attention so far. The goal is not only to optimize performance of some classifiers, but also the size of the training dataset must be reduced as well as the computational time of learning algorithms.

Edited Nearest Neighbor for Multi-Labelled data is an efficient editing method. The idea is, first, to classify all training instances using a k-NN rule, and, second, to eliminate erroneous instances based on a local criterion induced from the Hamming loss measure. The reduced set of instances is then used to classify unseen instances. We have demonstrated the effect of editing dataset on two learning algorithms: the EMLkNN and the Rank-SVM. This was illustrated through an example on synthetic data.

We applied our algorithm of editing to five real-world datasets from different domains of application: multimedia classification, bioinformatics and text categorization. Experimenting with these datasets, we observed that the learning algorithms (EMLkNN and Rank-SVM) with the editing datasets significantly outperformed the same algorithms on the initial datasets in terms of classification performance and computational costs. The explanation is that the editing datasets are distributed in more homogeneous clusters by reducing the

number of irrelevant instances. Learning from these new instances is faster with better generalization ability.

Note that the scalability of the presented approach depends on that of the k-NN algorithm. Several approaches exist to adapt the use of the k-NN on large datasets. For example, a simple approach is to use the idea of clustering in order to not to compute distances of a testing instance with respect to all training instances. It is clear that if the training data is *very noisy*, our method of edition may be a solution to reduce the number of training data by using only clean instances with a relatively small number with respect to the initial dataset.

Future research should consider applying the existing algorithm to other application domains, e.g., audio, video and images and showing extensive results on several classifiers, including for example decision trees [6], neural networks [45] and classifier chains [26], to investigate better the merit of editing in these settings. A very interesting idea will be to apply the edition on the training set using another method than the k-NN based one. Surely, this idea will not be straightforward. For example, if we decide to use the Rank SVM, we have to edit instances that are misclassified (since these instances increase the Hamming Loss), but it is well known that these instances (with non zero Lagrange multipliers) are involved in the construction of decision boundaries.

Table 5
Experimental results on the Emotions dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.209(4)	0.204(2)	0.206(3)	**0.194**(1)
One-Error$^-$	0.287(2)	0.297(3)	0.302(4)	**0.277**(1)
Coverage$^-$	1.881(2)	2.010(4)	1.896(3)	**1.847**(1)
Ranking loss$^-$	0.168(3)	0.220(4)	0.166(2)	**0.158**(1)
Average Precision$^+$	0.7994(2)	0.7959(4)	0.7993(3)	**0.8080**(1)
Accuracy$^+$	0.519(4)	**0.569**(1)	0.546(3)	0.561(2)
Precision$^+$	0.656(3)	**0.705**(1)	0.651(4)	0.690(2)
Recall$^+$	0.592(3)	**0.657**(1)	0.642(2)	0.642(2)
F1$^+$	0.596(4)	**0.648**(1)	0.621(3)	0.637(2)

+(-): the higher (smaller) the value, the better the performance.

Table 6
Experimental results on the Yeast dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.205(4)	0.202(3)	0.197(2)	**0.193**(1)
One-Error$^-$	0.261(4)	0.249(3)	**0.221**(1)	0.238(2)
Coverage$^-$	6.494(3)	6.577(4)	6.424(2)	**6.269**(1)
Ranking loss$^-$	0.188(3)	0.201(4)	0.167(2)	**0.165**(1)
Average precision$^+$	0.751(3)	0.751(4)	0.767(2)	**0.768**(1)
Accuracy$^+$	0.515(4)	0.529(2)	0.522(3)	**0.539**(1)
Precision$^+$	0.685(4)	0.689(3)	0.697(2)	**0.703**(1)
Recall$^+$	0.599(4)	0.618(3)	0.625(3)	**0.635**(1)
F1$^+$	0.613(4)	0.627(3)	0.628(3)	**0.641**(1)

+(-): the higher (smaller) the value, the better the performance.

Table 7
Experimental results on the Medical dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.018(3)	0.018(4)	0.012(2)	**0.011**(1)
One-Error$^-$	0.285(2)	0.291(3)	**0.141**(1)	**0.141**(1)
Coverage$^-$	3.541(3)	3.661(4)	**1.135**(1)	1.255(2)
Ranking loss$^-$	0.124(3)	0.126(4)	**0.015**(1)	0.018(2)
Average precision$^+$	0.779(3)	0.776(4)	0.897(2)	**0.898**(1)
Accuracy$^+$	0.559(4)	0.585(3)	0.688(2)	**0.726**(1)
Precision$^+$	0.617(4)	0.647(3)	0.744(2)	**0.781**(1)
Recall$^+$	0.569(4)	0.594(3)	0.718(2)	**0.754**(1)
F1$^+$	0.581(4)	0.608(3)	0.716(2)	**0.754**(1)

+(-): the higher (smaller) the value, the better the performance.

A Evaluation measures

As discussed in Section 3.2, Performance evaluation for multi-label learning systems differs from that of single-label classification. Let $\mathcal{H} : \mathbb{X} \to 2^{\mathcal{Y}}$ be a multi-label classifier that assigns a predicted label subset of $\mathcal{Y} = \{\omega_1, \ldots, \omega_Q\}$ to each instance $\mathbf{x} \in \mathbb{X}$, and let $f : \mathbb{X} \times \mathcal{Y} \to [0, 1]$ be the corresponding scoring function which gives a score for each label ω_q which in turn is interpreted as

Table 8
Experimental results on the Enron dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.057(3)	0.059(4)	0.055(2)	**0.053**(1)
One-Error$^-$	0.437(3)	0.478(4)	0.287(2)	**0.275**(1)
Coverage$^-$	21.959(3)	26.226(4)	13.758(2)	**12.772**(1)
Ranking loss$^-$	0.261(3)	0.395(4)	0.099(2)	**0.090**(1)
Average precision$^+$	0.568(3)	0.509(4)	0.619(2)	**0.647**(1)
Accuracy$^+$	0.303(4)	0.318(3)	0.398(2)	**0.436**(1)
Precision$^+$	0.473(4)	0.484(3)	0.574(2)	**0.587**(1)
Recall$^+$	0.340(4)	0.359(3)	0.495(2)	**0.556**(1)
F1$^+$	0.372(4)	0.390(3)	0.511(2)	**0.550**(1)

+(-): the higher (smaller) the value, the better the performance.

Table 9
Experimental results on the Webpage dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	0.065(4)	0.058(3)	0.043(2)	**0.042**(1)
One-Error$^-$	0.589(4)	0.558(3)	0.417(2)	**0.402**(1)
Coverage$^-$	12.164(3)	12.906(4)	5.080(2)	**4.169**(1)
Ranking loss$^-$	0.506(3)	0.545(4)	0.128(2)	**0.1000**(1)
Average precision$^+$	0.471(3)	0.470(4)	0.651(2)	**0.673**(1)
Accuracy$^+$	0.338(4)	0.364(3)	0.402(2)	**0.437**(1)
Precision$^+$	0.393(4)	0.426(3)	0.465(2)	**0.508**(1)
Recall$^+$	0.387(4)	0.391(3)	0.435(2)	**0.467**(1)
F1$^+$	0.371(4)	0.392(3)	0.432(2)	**0.469**(1)

+(-): the higher (smaller) the value, the better the performance.

the probability that ω_q is relevant. The function $f(.,.)$ can be transformed to a ranking function $rank_f(.,.)$ which maps the outputs of $f(\mathbf{x}, \omega)$ for any $\omega \in \mathcal{Y}$ to $\{\omega_1, \omega_2, \ldots, \omega_Q\}$ so that $f(\mathbf{x}_i, \omega_q) > f(\mathbf{x}_i, \omega_r)$ implies that $rank_f(\mathbf{x}_i, \omega_q) < rank_f(\mathbf{x}_i, \omega_r)$.

Given a set $\mathcal{S} = \{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_m, Y_m)\}$ of m test examples, the evaluation metrics of multi-label learning systems are divided into two groups: *prediction-based* and *ranking-based* metrics. *Prediction-based* measures are calculated

Table 10
Information about the used datasets.

	Number of instances in initial training data	Number of instances in edited training data	Editing Time (seconds)
Emotions	391	113	2.4
Yeast	1500	832	30.8
Medical	645	624	6.2
Enron	1123	861	9.2
Webpage	22000	13693	435.3

Table 11
Running Time (in Seconds) for learning and testing for the two methods.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Emotions	1.4	0.4	162.9	9.9
Yeast	12.4	10.1	$1.1 * 10^4$	$0.2 * 10^4$
Medical	7.3	3.7	$1.6 * 10^4$	$1.4 * 10^4$
Enron	18.8	8.2	$1.0 * 10^4$	$1.6 * 10^4$
Webpage	61.7	47.7	$2.1 * 10^4$ s $\simeq$ 14 h	$8.6 * 10^3$ s $\simeq$ 5.9 h

Table 12
Experimental results (mean±std) on the Emotions dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	$0.191 \pm 0.019\bullet$	0.146 ± 0.064	$0.192 \pm 0.022\bullet$	0.148 ± 0.0474
One-Error$^-$	$0.266 \pm 0.044\bullet$	0.188 ± 0.100	$0.256 \pm 0.081\bullet$	0.158 ± 0.0691
Coverage$^-$	$1.816 \pm 0.198\circ$	1.524 ± 0.402	$1.702 \pm 0.293\circ$	1.505 ± 0.4016
Ranking loss$^-$	$0.173 \pm 0.028\circ$	0.131 ± 0.061	$0.156 \pm 0.039\bullet$	0.100 ± 0.0495
Average precision$^+$	$0.799 \pm 0.030\bullet$	0.864 ± 0.054	$0.807 \pm 0.044\bullet$	0.874 ± 0.0477
Accuracy$^+$	$0.558 \pm 0.045\bullet$	0.681 ± 0.119	$0.541 \pm 0.049\bullet$	0.658 ± 0.0946
Precision$^+$	$0.688 \pm 0.052\bullet$	0.774 ± 0.095	$0.663 \pm 0.065\bullet$	0.757 ± 0.0805
Recall$^+$	$0.641 \pm 0.051\bullet$	0.772 ± 0.086	$0.654 \pm 0.062\bullet$	0.768 ± 0.1017
F1$^+$	$0.635 \pm 0.045\bullet$	0.748 ± 0.095	$0.626 \pm 0.052\bullet$	0.735 ± 0.0850

•(○): statistically significant (non-significant) difference of performance of the classification algorithm applied on the initial and the edited dataset, based on two-tailed paired t-test at 5% significance.

based on the average difference of the actual and the predicted set of labels over all test examples. *Ranking-based* metrics evaluate the label ranking quality depending on the scoring function $f(.,.)$.

Table 13
Experimental results (mean±std) on the Yeast dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	$0.201 \pm 0.010\bullet$	0.168 ± 0.041	$0.195 \pm 0.006\bullet$	0.157 ± 0.0384
One-Error$^-$	$0.242 \pm 0.027\bullet$	0.170 ± 0.074	$0.216 \pm 0.032\circ$	0.165 ± 0.0724
Coverage$^-$	$6.481 \pm 0.263\bullet$	5.727 ± 0.800	$6.336 \pm 0.236\bullet$	5.496 ± 0.7416
Ranking loss$^-$	$0.186 \pm 0.015\bullet$	0.139 ± 0.053	$0.165 \pm 0.007\bullet$	0.116 ± 0.0451
Average precision$^+$	$0.757 \pm 0.018\bullet$	0.813 ± 0.059	$0.773 \pm 0.013\bullet$	0.829 ± 0.0587
Accuracy$^+$	$0.524 \pm 0.021\bullet$	0.604 ± 0.074	$0.529 \pm 0.016\bullet$	0.611 ± 0.0763
Precision$^+$	$0.682 \pm 0.024\bullet$	0.734 ± 0.060	$0.695 \pm 0.016\bullet$	0.755 ± 0.0571
Recall$^+$	$0.614 \pm 0.023\bullet$	0.705 ± 0.076	$0.633 \pm 0.022\bullet$	0.713 ± 0.0795
F1$^+$	$0.621 \pm 0.021\bullet$	0.697 ± 0.068	$0.634 \pm 0.011\bullet$	0.707 ± 0.0692

•(○): statistically significant (non-significant) difference of performance of the classification algorithm applied on the initial and the edited dataset, based on two-tailed paired t-test at 5% significance.

Table 14
Experimental results (mean±std) on the Medical dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	$0.017 \pm 0.003\circ$	0.015 ± 0.002	$0.011 \pm 0.002\circ$	0.010 ± 0.001
One-Error$^-$	$0.277 \pm 0.068\circ$	0.242 ± 0.028	$0.137 \pm 0.033\circ$	0.118 ± 0.034
Coverage$^-$	$3.356 \pm 1.164\circ$	4.129 ± 0.881	$1.070 \pm 0.370\circ$	0.864 ± 0.284
Ranking loss$^-$	$0.108 \pm 0.034\bullet$	0.151 ± 0.027	$0.014 \pm 0.007\circ$	0.011 ± 0.005
Average precision$^+$	$0.784 \pm 0.047\circ$	0.796 ± 0.019	$0.906 \pm 0.022\circ$	0.915 ± 0.021
Accuracy$^+$	$0.592 \pm 0.064\bullet$	0.642 ± 0.034	$0.719 \pm 0.042\bullet$	0.769 ± 0.030
Precision$^+$	$0.654 \pm 0.065\bullet$	0.705 ± 0.033	$0.761 \pm 0.049\bullet$	0.810 ± 0.034
Recall$^+$	$0.611 \pm 0.061\bullet$	0.668 ± 0.038	$0.757 \pm 0.041\bullet$	0.819 ± 0.032
F1$^+$	$0.619 \pm 0.063\bullet$	0.672 ± 0.034	$0.746 \pm 0.044\bullet$	0.799 ± 0.030

•(○): statistically significant (non-significant) difference of performance of the classification algorithm applied on the initial and the edited dataset, based on two-tailed paired t-test at 5% significance.

A.1 Prediction-based measures

Hamming loss: The hamming loss metric for the set of labels is defined as the fraction of labels whose relevance is incorrectly predicted:

$$\mathcal{H}Loss(\mathcal{H}, \mathcal{S}) = \frac{1}{m} \sum_{i=1}^{m} \frac{|Y_i \triangle \widehat{Y}_i|}{Q}, \tag{A.1}$$

where $\triangle$ denotes the symmetric difference between two sets.

Accuracy: The accuracy metric gives an average degree of similarity between the predicted and the ground truth label sets:

$$\mathcal{A}ccuracy(\mathcal{H}, \mathcal{S}) = \frac{1}{m} \sum_{i=1}^{m} \frac{|Y_i \cap \widehat{Y}_i|}{|Y_i \cup \widehat{Y}_i|}. \tag{A.2}$$

Table 15
Experimental results (mean±std) on the Enron dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	$0.062 \pm 0.010\circ$	0.057 ± 0.004	$0.056 \pm 0.014\circ$	0.049 ± 0.003
One-Error$^-$	$0.571 \pm 0.098\bullet$	0.390 ± 0.075	$0.294 \pm 0.111\circ$	0.215 ± 0.049
Coverage$^-$	$25.645 \pm 6.323\circ$	23.242 ± 2.423	$13.937 \pm 3.234\bullet$	11.476 ± 1.167
Ranking loss$^-$	$0.334 \pm 0.086\circ$	0.294 ± 0.058	$0.098 \pm 0.024\bullet$	0.072 ± 0.012
Average precision$^+$	$0.467 \pm 0.056\circ$	0.577 ± 0.052	$0.614 \pm 0.087\bullet$	0.702 ± 0.035
Accuracy$^+$	$0.165 \pm 0.058\bullet$	0.366 ± 0.048	$0.380 \pm 0.125\bullet$	0.486 ± 0.040
Precision$^+$	$0.334 \pm 0.106\bullet$	0.542 ± 0.059	$0.575 \pm 0.097\bullet$	0.660 ± 0.038
Recall$^+$	$0.184 \pm 0.066\bullet$	0.421 ± 0.054	$0.466 \pm 0.110\bullet$	0.599 ± 0.044
F1$^+$	$0.219 \pm 0.075\bullet$	0.448 ± 0.054	$0.494 \pm 0.105\bullet$	0.602 ± 0.040

•(○): statistically significant (non-significant) difference of performance of the classification algorithm applied on the initial and the edited dataset, based on two-tailed paired t-test at 5% significance.

Table 16
Experimental results (mean±std) on the Webpage dataset.

	EMLkNN		Rank-SVM	
	Before Editing	After Editing	Before Editing	After Editing
Hamming loss$^-$	$0.063 \pm 0.001\bullet$	0.050 ± 0.007	0.042 ± 0.005	$\mathbf{0.041 \pm 0.004}\circ$
One-Error$^-$	$0.564 \pm 0.007\bullet$	0.523 ± 0.059	0.399 ± 0.036	$\mathbf{0.394 \pm 0.035}\circ$
Coverage$^-$	$12.124 \pm 0.149\bullet$	10.313 ± 1.531	5.253 ± 0.397	$\mathbf{4.321 \pm 0.687}\bullet$
Ranking loss$^-$	$0.494 \pm 0.006\bullet$	0.415 ± 0.064	0.132 ± 0.009	$\mathbf{0.108 \pm 0.015}\bullet$
Average precision$^+$	$0.485 \pm 0.004\bullet$	0.532 ± 0.057	0.631 ± 0.031	$\mathbf{0.678 \pm 0.028}\bullet$
Accuracy$^+$	$0.359 \pm 0.004\circ$	0.372 ± 0.063	0.402 ± 0.028	$\mathbf{0.440 \pm 0.039}\bullet$
Precision$^+$	$0.417 \pm 0.004\circ$	0.425 ± 0.061	0.467 ± 0.032	$\mathbf{0.500 \pm 0.037}\bullet$
Recall$^+$	$0.402 \pm 0.006\circ$	0.396 ± 0.062	0.439 ± 0.030	$\mathbf{0.486 \pm 0.040}\bullet$
F1$^+$	$0.391 \pm 0.005\circ$	0.397 ± 0.062	0.434 ± 0.030	$\mathbf{0.474 \pm 0.039}\bullet$

+(-): the higher (smaller) the value, the better the performance.
•(○): statistically significant (non-significant) difference of performance of the classification algorithm applied on the initial and the edited dataset, based on two-tailed paired t-test at 5% significance.

Precision: The precision metric computes the proportion of true positive predictions:

$$\mathcal{P}recision(\mathcal{H}, \mathcal{S}) = \frac{1}{m} \sum_{i=1}^{m} \frac{|Y_i \cap \widehat{Y}_i|}{|\widehat{Y}_i|}. \tag{A.3}$$

Recall: This metric estimates the proportion of true labels that have been predicted as positives:

$$\mathcal{R}ecall(\mathcal{H}, \mathcal{S}) = \frac{1}{m} \sum_{i=1}^{m} \frac{|Y_i \cap \widehat{Y}_i|}{|Y_i|}. \tag{A.4}$$

F1-measure: $F1$ measure is defined as the harmonic mean of precision and

Table 17
P-values on the Emotions dataset.

	t-test		Friedman test	
	EMLkNN	Rank-SVM	EMLkNN	Rank-SVM
Hamming loss$^-$	0.048	0.016	0.058	0.206
One-Error$^-$	0.037	0.009	0.527	0.011
Coverage$^-$	0.054	0.225	0.206	1
Ranking loss$^-$	0.061	0.011	0.058	0.206
Average precision$^+$	0.004	0.004	0.011	0.011
Accuracy$^+$	0.007	0.003	0.011	0.058
Precision$^+$	0.021	0.010	0.011	0.011
Recall$^+$	0.001	0.007	0.011	0.206
F1$^+$	0.003	0.003	0.011	0.058

+(-): the higher (smaller) the value, the better the performance.

Table 18
P-values on the Yeast dataset.

	t-test		Friedman test	
	EMLkNN	Rank-SVM	EMLkNN	Rank-SVM
Hamming loss$^-$	0.023	0.007	0.206	0.058
One-Error$^-$	0.011	0.059	0.206	0.058
Coverage$^-$	0.011	0.003	0.058	0.058
Ranking loss$^-$	0.015	0.003	0.058	0.051
Average precision$^+$	0.010	0.008	0.206	0.058
Accuracy$^+$	0.004	0.004	0.058	0.058
Precision$^+$	0.020	0.005	0.206	0.011
Recall$^+$	0.002	0.007	0.508	0.058
F1$^+$	0.004	0.004	0.058	0.058

+(-): the higher (smaller) the value, the better the performance.

recall. It is calculated as:

$$\mathcal{F}1(\mathcal{H},\mathcal{S}) = \frac{1}{m}\sum_{i=1}^{m}\frac{2|Y_i \cap \widehat{Y}_i|}{|Y_i| + |\widehat{Y}_i|}. \tag{A.5}$$

Note that the smaller the value of the Hamming loss, the better the performance. For the other metrics, higher values correspond to better classification quality.

Table 19
P-values on the Medical dataset.

	t-test		Friedman test	
	EMLkNN	Rank-SVM	EMLkNN	Rank-SVM
Hamming loss$^-$	0.105	0.136	0.527	0.527
One-Error$^-$	0.144	0.224	0.058	1
Coverage$^-$	0.111	0.178	0.206	0.058
Ranking loss$^-$	0.006	0.325	0.058	0.058
Average precision$^+$	0.476	0.336	1	1
Accuracy$^+$	0.043	0.007	0.206	0.058
Precision$^+$	0.04	0.021	0.058	0.527
Recall$^+$	0.022	0.002	0.206	0.011
F1$^+$	0.032	0.005	0.058	0.058

+(-): the higher (smaller) the value, the better the performance.

Table 20
P-values on the Enron dataset.

	t-test		Friedman test	
	EMLkNN	Rank-SVM	EMLkNN	Rank-SVM
Hamming loss$^-$	0.154	0.112	0.058	0.011
One-Error$^-$	0	0.054	0.011	0.058
Coverage$^-$	0.276	0.036	0.206	0.011
Ranking loss$^-$	0.232	0.007	0.206	0.011
Average precision$^+$	0	0.008	0.011	0.011
Accuracy$^+$	0	0.019	0.002	0.011
Precision$^+$	0	0.018	0.011	0.011
Recall$^+$	0	0.002	0.002	0.011
F1$^+$	0	0.007	0.001	0.011

+(-): the higher (smaller) the value, the better the performance.

A.2 Ranking-based measures

One-error: This metric computes how many times the top-ranked label is not in the true set of labels of the instance, and it ignores the relevancy of all other labels.

$$OErr(f, \mathcal{S}) = \frac{1}{m}\sum_{i=1}^{m}\langle[\arg\max_{\omega \in Y} f(\mathbf{x}_i, \omega)] \notin Y_i\rangle, \tag{A.6}$$

Table 21
P-values on the Web dataset.

	t-test		Friedman test	
	EMLkNN	Rank-SVM	EMLkNN	Rank-SVM
Hamming loss$^-$	0	0.537	0.002	0.206
One-Error$^-$	0.042	0.569	0.206	0.527
Coverage$^-$	0.002	0	0.058	0.002
Ranking loss$^-$	0.001	0	0.011	0.001
Average precision$^+$	0.018	0.028	0.206	0.058
Accuracy$^+$	0.500	0.117	0.058	0.058
Precision$^+$	0.675	0.281	0.206	0.206
Recall$^+$	0.744	0.054	0.011	0.058
F1$^+$	0.757	0.117	0.058	0.058

+(-): the higher (smaller) the value, the better the performance.

where for any proposition H, $\langle H \rangle$ equals to 1 if H holds and 0 otherwise. Note that, for single-label classification problems, the One Error is identical to ordinary classification error.

Coverage: Coverage computes the average of how far we need to move down the ranked label list in order to cover all the labels assigned to a test instance.

$$\mathcal{C}ov(f, \mathcal{S}) = \frac{1}{m} \sum_{i=1}^{m} \max_{\omega \in Y_i} rank_f(\mathbf{x}_i, \omega) - 1. \tag{A.7}$$

Ranking loss: This metric computes the number of times that an incorrect label is ranked higher than a correct label.

$$\mathcal{R}Loss(f, \mathcal{S}) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{|Y_i||\overline{Y}_i|} |(\omega_q, \omega_r) \in Y_i \times \overline{Y}_i \backslash f(\mathbf{x}_i, \omega_q) \leq f(\mathbf{x}_i, \omega_r)| \tag{A.8}$$

where $\overline{Y}_i$ is the complementary set of Y_i in $\mathcal{Y}$.

Average precision: This metric evaluates the average fraction of labels ranked above a particular label $\omega \in Y_i$ which are actually in Y_i.

$$\mathcal{A}vPrec(f, \mathcal{S}) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{|Y_i|} \sum_{\omega_q \in Y_i} \frac{|\{\omega_r \in Y_i\} \backslash rank_f(\mathbf{x}_i, \omega_r) \leq rank_f(\mathbf{x}_i, \omega_q)|}{rank_f(\mathbf{x}_i, \omega_q)}. \tag{A.9}$$

Note that $AvPrec(f, \mathcal{S}) = 1$ means that the labels are perfectly ranked. For the other metrics, smaller values correspond to a better label ranking quality.

B Multi-labeled dataset statistics

Given a multi-labeled dataset $\mathcal{D} = \{(\mathbf{x}_i, Y_i), i = 1, \ldots, n\}$ with $\mathbf{x}_i \in \mathbb{X}$ and $Y_i \subseteq \mathcal{Y}$, this dataset can be measured by the number of instances (n), the number of attributes in the input space, and the number of labels (Q). In the following, we review some statistics about the multi-labeled dataset $\mathcal{D}$ [36].

Label Cardinality: The Label Cardinality (LCard) of $\mathcal{D}$ is the average number of labels per instance. Label cardinality is calculated as

$$LCard(\mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} |Y_i| \tag{B.1}$$

Label Density: The Label Density (LDen) of $\mathcal{D}$ is defined as the average number of labels per instance divided by the total number of labels Q. Label density is calculated as:

$$LDen(\mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} \frac{|Y_i|}{Q} \tag{B.2}$$

Both metrics indicate the number of alternative labels that characterize the examples of a multi-labeled dataset. Label cardinality is independent of the total number of labels in the classification problem, while label density takes into consideration the total number of labels. Two datasets with the same label cardinality but with different label densities may present different properties that influence the performance of the multi-label classification methods.

Distinct Label sets: The Distinct Label sets (DL) counts the number of label sets that are unique across the total number of examples. Distinct label sets is given by:

$$DL(\mathcal{D}) = |\{Y_i \subseteq \mathcal{Y} | \exists\ \mathbf{x}_i \in \mathbb{X}\ :\ (\mathbf{x}_i, Y_i) \in \mathcal{D}\}| \tag{B.3}$$

This measure gives an idea of the regularity of the labeling scheme.

[3] `http://www.wormbook.org/chapters/www_genomclassprot/genomclassprot.html`

References

[1] Barbedo, J.G.A., Lopes, A.: Automatic Genre Classification of Musical Signals. EURASIP Journal on Advances in Signal Processing **2007**(1), 064,960 (2007). DOI 10.1155/2007/64960

[2] Blockeel, H., De Raedt, L., Ramon, J.: Top-down induction of clustering trees. In: the 15th International Conference on Machine Learning, pp. 55–63. Morgan Kaufmann (1998)

[3] Brighton, H., Mellish, C.: Advances in Instance Selection for Instance-Based Learning Algorithms. Data Mining and Knowledge Discovery **6**(2), 153–172 (2002). DOI 10.1023/A:1014043630878

[4] Brodley, C.E., Friedl, M.A.: Identifying mislabeled training data. Journal Of Artificial Intelligence Research **11**, 131–167 (1999). DOI 10.1613/jair.606

[5] de Carvalho, A., Freitas, A.A.: A Tutorial on Multi-label Classification Techniques. In: Foundations of Computational Intelligence Volume 5, Studies in Computational Intelligence, pp. 177–195. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). DOI 10.1007/978-3-642-01536-6_8

[6] Clare, A., King, R.D.: Knowledge Discovery in Multi-Label Phenotype Data. In: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD '01), vol. 2168, pp. 42–53. Springer-Verlag, London, UK, Baden-Wurttemberg, Germany (2001)

[7] Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory **13**(1), 21–27 (1967). DOI 10.1109/TIT.1967.1053964

[8] Dasarathy, B.V.: Nearest neighbor (NN) norms: NN Pattern Classification Techniques. IEEE Computer Society Press (1991). DOI ieeecomputersociety.org/10.1109/2.84880

[9] Denoeux, T.: A k-nearest neighbor classification rule based on Dempster-Shafer Theory. IEEE Transactions on Systems, Man, and Cybernetics **25**(05), 804–813 (1995)

[10] Denoeux, T., Younes, Z., Abdallah, F.: Representing uncertainty on set-valued variables using belief functions. Artificial Intelligence **174**(7-8), 479–499 (2010). DOI 10.1016/j.artint.2010.02.002

[11] Devijver, P.A.: On the editing rate of the Multiedit algorithm. Pattern Recognition Letters **4**(1), 9–12 (1986). DOI 10.1016/0167-8655(86)90066-8

[12] Elisseeff, A., Weston, J.: Kernel methods for Multi-labelled classification and Categorical regression problems. In: Advances in Neural Information Processing Systems 14, vol. 14, pp. 681–687. Biowulf Technologies, MIT Press (2001)

[13] García, S., Derrac, J., Cano, J.R., Herrera, F.: Prototype selection for nearest neighbor classification: taxonomy and empirical study. IEEE transactions on

pattern analysis and machine intelligence **34**(3), 417–35 (2012). DOI 10.1109/TPAMI.2011.142

[14] Garcia, S., Herrera, F.: An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons. Journal of Machine Learning Research **9**, 2677–2694 (2008)

[15] Guan, D., Yuan, W., Lee, Y.K., Lee, S.: Nearest neighbor editing aided by unlabeled data. Information Sciences **179**(13), 2273–2282 (2009). DOI 10.1016/j.ins.2009.02.011

[16] Hattori, K., Takahashi, M.: A new edited k-nearest neighbor rule in the pattern classification problem. Pattern Recognition **33**(3), 521–528 (2000). DOI 10.1016/S0031-3203(99)00068-0

[17] Jin, B., Muller, B., Zhai, C., Lu, X.: Multi-label literature classification based on the Gene Ontology graph. BMC bioinformatics **9**, 525 (2008). DOI 10.1186/1471-2105-9-525

[18] Kanj, S., Abdallah, F., Denoeux, T.: Purifying training data to improve performance of multi-label classification algorithms. In: Proceedings of the 15th Int. Conf. on Information Fusion (FUSION 2012), pp. 1784–1792. IEEE, Singapore (2012)

[19] Koplowitz, J., Brown, T.A.: On the relation of performance to editing in nearest neighbor rules. Pattern Recognition **13**(3), 251–255 (1981). DOI 10.1016/0031-3203(81)90102-3

[20] Li, Y., Hu, Z., Cai, Y., Zhang, W.: Support vector based prototype selection method for nearest neighbor rules. In: Proceedings of the first international conference on Advances in Natural Computation, pp. 528–535. Springer Berlin Heidelberg, Changsha, China (2005). DOI 10.1007/11539087_68

[21] Madjarov, G., Kocev, D., Gjorgjevikj, D., Džeroski, S.: An extensive experimental comparison of methods for multi-label learning. Pattern Recognition **45**(9), 3084–3104 (2012). DOI 10.1016/j.patcog.2012.03.004

[22] Pavlidis, P., Grundy, W.N.: Combining microarray expression data and phylogenetic profiles to learn gene functional categories using support vector machines. Tech. rep., Department of Computer Science, Columbia University, New York (2000)

[23] Pestian, J.P., Brew, C., Matykiewicz, P., Hovermale, D.J., Johnson, N., Cohen, K.B., Duch, W.: A Shared Task Involving Multi-label Classification of Clinical Free Text. In: Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing (BioNLP '07), vol. 1, pp. 97–104. Association for Computational Linguistics, Prague, Czech Republic (2007)

[24] Pkalska, E., Duin, R.P., Paclík, P.: Prototype selection for dissimilarity-based classifiers. Pattern Recognition **39**(2), 189–208 (2006). DOI 10.1016/j.patcog.2005.06.012

[25] Qi, G.J., Hua, X.S., Rui, Y., Tang, J., Mei, T., Zhang, H.J.: Correlative multi-label video annotation. In: Proceedings of the 15th international conference on Multimedia - MULTIMEDIA '07, p. 17. ACM Press, Augsburg, Germany (2007). DOI 10.1145/1291233.1291245

[26] Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning **85**(3), 333–359 (2011). DOI 10.1007/s10994-011-5256-5

[27] Sánchez, J., Pla, F., Ferri, F.: Prototype selection for the nearest neighbour rule through proximity graphs. Pattern Recognition Letters **18**(6), 507–513 (1997). DOI 10.1016/S0167-8655(97)00035-4

[28] Schapire, R.E., Singer, Y.: Improved Boosting Algorithms Using Confidence-rated Predictions. Machine learning **37**(3), 297–336 (1999). DOI 10.1023/A:1007614523901

[29] Schapire, R.E., Singer, Y.: BoosTexter : A Boosting-based System for Text Categorization. Machine Learning **39**(2-3), 135–168 (2000). DOI 10.1023/A:1007649029923

[30] Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys **34**(1), 1–47 (2002). DOI 10.1145/505282.505283

[31] Shetty, J., Adibi, J.: The Enron Email Dataset Database Schema and Brief Statistical Report. Tech. rep., Information Sciences Institute Technical Report, University of Southern California (2004)

[32] Tahir, M.A., Kittler, J., Bouridane, A.: Multilabel classification using heterogeneous ensemble of multi-label classifiers. Pattern Recognition Letters **33**(5), 513–523 (2012). DOI 10.1016/j.patrec.2011.10.019

[33] Tang, L., Rajan, S., Narayanan, V.K.: Large scale multi-label classification via metalabeler. In: Proceedings of the 18th international conference on World wide web (WWW '09), p. 211. ACM Press, Madrid, Spain (2009). DOI 10.1145/1526709.1526738

[34] Tomek, I.: Two Modifications of CNN. IEEE Transactions on Systems, Man, and Cybernetics **6**(11), 769–772 (1976). DOI 10.1109/TSMC.1976.4309452

[35] Trohidis, K., Tsoumakas, G., Kalliris, G., Vlahavas, I.: Multi-label classification of music into emotions. In: Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR '08), pp. 325–330. Philadelphia, PA, USA (2008)

[36] Tsoumakas, G., Katakis, I.: Multi-Label Classification : An Overview. International Journal of data warehousing & mining **3**(3), 1–13 (2007)

[37] Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining Multi-label Data. In: O. Maimon, L. Rokach (eds.) Data Mining and Knowledge Discovery Handbook, pp. 667–685. Springer US, Thessaloniki, Greece (2010). DOI 10.1007/978-0-387-09823-4_34

[38] Tsoumakas, G., Katakis, I., Vlahavas, I.: Random k-Labelsets for Multilabel Classification. IEEE Transactions on Knowledge and Data Engineering **23**(7), 1079–1089 (2011). DOI 10.1109/TKDE.2010.164

[39] Van Hulse, J., Khoshgoftaar, T.: Knowledge discovery from imbalanced and noisy data. Data & Knowledge Engineering **68**(12), 1513–1542 (2009). DOI 10.1016/j.datak.2009.08.005

[40] Wilson, D.L.: Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Transactions on Systems, Man, and Cybernetics **2**(3), 408–421 (1972). DOI 10.1109/TSMC.1972.4309137

[41] Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G.J., Ng, A., Liu, B., Yu, P.S., Zhou, Z.H., Steinbach, M., Hand, D.J., Steinberg, D.: Top 10 algorithms in data mining. Knowledge and Information Systems **14**(1), 1–37 (2007). DOI 10.1007/s10115-007-0114-2

[42] Xu, J.: An extended one-versus-rest support vector machine for multi-label classification. Neurocomputing **74**(17), 3114–3124 (2011). DOI 10.1016/j.neucom.2011.04.024

[43] Xu, J.: An efficient multi-label support vector machine with a zero label. Expert Systems with Applications **39**(5), 4796–4804 (2012). DOI 10.1016/j.eswa.2011.09.138

[44] Younes, Z., Abdallah, F., Denoeux, T., Snoussi, H.: A Dependent Multilabel Classification Method Derived from the -Nearest Neighbor Rule. EURASIP Journal on Advances in Signal Processing **2011**(1), 645,964 (2011). DOI 10.1155/2011/645964

[45] Zhang, M.L., Zhou, Z.H.: Multilabel neural networks with applications to functional genomics and text categorization. IEEE Trans. on Knowl. and Data Eng. **18**(10), 1338–1351 (2006). DOI 10.1109/TKDE.2006.162

[46] Zhang, M.L., Zhou, Z.h.: ML-KNN: A lazy learning approach to multi-label learning. Pattern Recognition **40**(7), 2038–2048 (2007). DOI 10.1016/j.patcog.2006.12.019

[47] Zhou, Z.H., Zhang, M.L.: Multi-instance multi-label learning with application to scene classification. In: the Twenty-Second Conference on Artificial Intelligence (AAAI), vol. 40, pp. 1609–1616. MIT Press, Vancouver, Canada (2007). DOI 10.1016/j.patcog.2006.12.019

[48] Zouhal, L., Denoeux, T.: An evidence-theoretic k-NN rule with parameter optimization. IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews) **28**(2), 263–271 (1998). DOI 10.1109/5326.669565