Multi-Label Quantification (Extended Abstract)

Alejandro Moreo[1], Manuel Francisco[2], Fabrizio Sebastiani[1]

[1]Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
E-mail: {alejandro.moreo,fabrizio.sebastiani}@isti.cnr.it

[2]Department of Computer Science and Artificial Intelligence
University of Granada
18071 Granada, Spain
E-mail: francisco@decsai.ugr.es

Abstract. While many quantification methods have been proposed in the past for binary problems and, to a lesser extent, single-label multiclass problems, the multi-label setting (i.e., the scenario in which the classes of interest are not mutually exclusive) remains by and large unexplored. A straightforward solution to the multi-label quantification problem could simply consist of recasting the problem as a set of independent binary quantification problems. Such a solution is simple but naïve, since the independence assumption upon which it rests is, in most cases, not satisfied. In these cases, knowing the relative frequency of one class could be of help in determining the prevalence of other related classes. We propose the first truly multi-label quantification methods, i.e., methods for inferring estimators of class prevalence values that strive to leverage the stochastic dependencies among the classes of interest in order to predict their relative frequencies more accurately. We show empirical evidence that natively multi-label solutions outperform the naïve approaches by a large margin.

1 Introduction

One important setting which remains to a large extent unexplored in the quantification literature is *multi-label quantification* (MLQ), the scenario in which every datapoint may belong to zero, one, or several classes at the same time; in this paper we set out to analyze MLQ systematically.

We start by noting that, since quantification systems are expected to be robust to prior probability shift, we need to test them against datasets exhibiting substantial amounts of shift. Our first contribution is the first experimental protocol specifically designed for multi-label quantification, a protocol that guarantees that the data MLQ systems are tested against do comply with the above desideratum.

We carry on by noting that a trivial solution for MLQ could simply consist of training one independent binary quantifier for each of the classes in the

codeframe. However, such a solution is arguably a "naïve" one, as it assumes the classes to be independent of each other, and thus does not attempt to leverage the *class-class correlations*, i.e., the stochastic dependencies that may exist among different classes. We show empirical evidence that multi-label quantifiers built according to this naïve intuition yield suboptimal performance, and that this happens independently of the method used for training the binary quantifiers.

We then move on to studying different possible strategies for tackling MLQ, and subdivide these strategies in four groups, based on their way of addressing (if at all) the multi-label nature of the problem. While the first two groups can be instantiated by using already available techniques, the other two cannot, since this would require "aggregation" techniques (see Section 4) that leverage the stochastic relations between classes, and no such method has been proposed before. We indeed propose two such methods, called RQ and LPQ. Extensive experiments that we have carried out using 15 publicly available datasets show that, when working in combination with a classifier that itself leverages the above-mentioned stochastic relations, LPQ and (especially) RQ outperform all other MLQ techniques. The code to reproduce all our experiments is available at `https://github.com/manuel-francisco/quapy-ml/`. An extended version of this paper, available at `https://dl.acm.org/doi/10.1145/3606264` and forthcoming as (Moreo et al., 2024), reports all the experimental results, that we here omit for reasons of space.

2 Notation and Definitions

In this paper we use the following notation. By $\mathbf{x}$ we indicate a datapoint drawn from a domain $\mathcal{X}$ of datapoints, while by y we indicate a class drawn from a finite, predefined set of classes (also known as a *codeframe*) $\mathcal{Y} = \{y_1, ..., y_n\}$, with n the number of classes of interest. Symbol σ denotes a *sample*, i.e., a non-empty set of (labelled or unlabelled) datapoints drawn from $\mathcal{X}$. By $p_\sigma(y)$ we indicate the *true* prevalence of class y in sample σ, by $\hat{p}_\sigma(y)$ we indicate an *estimate* of this prevalence, and by $\hat{p}^q_\sigma(y)$ we indicate the estimate of this prevalence obtained by means of quantification method q. We will denote by $\mathbf{p} = (p_1, \ldots, p_n)$ a real-valued vector. When $\mathbf{p}$ is a vector of class prevalence values, then p_i is short for $p_\sigma(y_i)$.

We first formalize the SLQ problem (Section 2.1) and then propose a definition of the MLQ problem (Section 2.2).

2.1 Single-Label Codeframes

In single-label problems, each datapoint $\mathbf{x}$ belongs to one and only one class in $\mathcal{Y}$. We denote a datapoint with its true class label as a pair $(\mathbf{x}, y)$, indicating that $y \in \mathcal{Y}$ is the true label of $\mathbf{x} \in \mathcal{X}$. We represent a set of k datapoints as $\{(\mathbf{x}^{(i)}, y^{(i)})_{i=1}^k : \mathbf{x}^{(i)} \in \mathcal{X}, y^{(i)} \in \mathcal{Y}\}$. By L we denote a collection of labelled datapoints, that we typically use as a training set, while by U we denote a collection of unlabelled datapoints, that we typically use for testing purposes.

We define a *single-label hard classifier* as a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ i.e., a predictor of the class attributed to a datapoint. We will instead take a *single-label soft classifier* to be a function $s : \mathcal{X} \rightarrow \Delta^{n-1}$ with Δ^{n-1} the unit (n-1)-simplex (aka *probability simplex* or *standard simplex*) defined as $\Delta^{n-1} = \{(p_1, \ldots, p_n) \mid p_i \in [0, 1], \sum_{i=1}^{n} p_i = 1\}$ i.e., as the domain of all vectors representing probability distributions over $\mathcal{Y}$. We define a *single-label quantifier* as a function $q : 2^{\mathcal{X}} \rightarrow \Delta^{n-1}$ i.e., a function mapping samples drawn from $\mathcal{X}$ into probability distributions over $\mathcal{Y}$.

Note that, despite the fact that the codomains of soft classifiers and quantifiers are the same, in the former case the i-th component of $s(\mathbf{x})$ denotes the posterior probability $\Pr(y_i|\mathbf{x})$, i.e., the probability that $\mathbf{x}$ belongs to class y_i as estimated by s, while in the latter case it denotes the class prevalence value $p_\sigma(y_i)$ as estimated by q.

By $d(\mathbf{p}, \hat{\mathbf{p}})$ we denote an evaluation measure for SLQ; these measures are typically *divergences*, i.e., functions that measure the amount of discrepancy between two probability distributions. Everything we say for single-label problems applies to the binary case as well, since the latter is the special case of the former in which $n = 2$, with one class typically acting as the "positive class", and the other as the "negative class".

2.2 Multi-Label Codeframes

In multi-label problems each datapoint $\mathbf{x}$ can belong to zero, one, or more than one class in $\mathcal{Y}$; as a result, the sum $\sum_{i=1}^{n} p_i$ may be different from 1. We denote a datapoint with its true labels as a pair $(\mathbf{x}, Y)$, in which $Y \subseteq \mathcal{Y}$ is the set of true labels assigned to $\mathbf{x} \in \mathcal{X}$. A multi-label collection with k datapoints is represented as $\{(\mathbf{x}^{(i)}, Y^{(i)})_{i=1}^{k} : \mathbf{x}^{(i)} \in \mathcal{X}, Y^{(i)} \subseteq \mathcal{Y}\}$. We define a *multi-label hard classifier* as a function $h : \mathcal{X} \rightarrow 2^{\mathcal{Y}}$ i.e., as a classifier that can assign zero, one, or more than one label to each datapoint, while we define a *multi-label soft classifier* as a function $s : \mathcal{X} \rightarrow [0, 1]^n$ Note that, unlike in the single-label case, the codomain of function s is not a probability simplex, since the sum $\sum_{i=1}^{n} p_i$ may be different from 1, but the set of all real-valued vectors $(p_1, \ldots, p_n)$ such that $p_i \in [0, 1]$.

We define a *multi-label quantifier* as a function $q : 2^{\mathcal{X}} \rightarrow [0, 1]^n$ i.e., a function mapping samples from $\mathcal{X}$ into vectors of n class prevalence values, where, differently from the single-label multiclass case, the class prevalence values in a vector do not need to sum up to 1.

3 An Evaluation Protocol for Testing Multi-Label Quantifiers

For the evaluation of quantifiers, researchers often use the same datasets that are elsewhere used for testing classifiers. On one hand this looks natural, because both classification and quantification deal with datapoints that belong to classes in a given codeframe. On the other hand this looks problematic, since

classification deals with estimating class labels for individual datapoints while quantification deals with estimating class prevalence values for *samples* (sets) of such datapoints. Simply estimating the accuracy of a quantifier on the entire test set of a dataset used for classification purposes (hereafter: a "classification dataset") would not be enough, since this would be a single prediction only, which would be akin to testing a classifier on a single datapoint only. As a result, it is customary to generate a dataset to be used for quantification purposes (a "quantification dataset") from a classification dataset by extracting from the test set of the latter a number of samples than will form the test set of the quantification dataset. Exactly how these samples are extracted is specified by an *evaluation protocol*. Different evaluation protocols for the binary case (Esuli and Sebastiani, 2015; Forman, 2005), for the single-label multiclass case (Esuli et al., 2022), and for the ordinal case (Bunse et al., 2022), have been proposed in the quantification literature.

For the binary case, the most widely adopted protocol is the so-called *artificial prevalence protocol* (APP) (Forman, 2005). The APP consists of extracting, from a set of test datapoints, many samples at controlled prevalence values. The APP takes four parameters as input: the unlabelled collection U, the sample size k, the number of samples m to draw for each predefined vector of prevalence values, and a grid of prevalence values $\mathbf{g}$ (e.g., $\mathbf{g} = (0.0, 0.1, \ldots, 0.9, 1.0)$). We then generate all the vectors $\mathbf{p} = (p(\oplus), p(\ominus))$ of $n = 2$ prevalence values consisting of combinations of values from the grid $\mathbf{g}$ that represent valid distributions (i.e., such that the elements in $\mathbf{p}$ sum up to 1). For each such prevalence vector, we then draw m different samples of k elements each, which become the elements of our test set. The APP thus confronts the quantifier with samples characterized by class prevalence values very different from the ones seen during training, and can thus test the robustness of the quantifiers to the presence of prior probability shift. This protocol is, by far, the most popular one in the quantification literature (see, e.g., (Card and Smith, 2018; Esuli et al., 2018; Fernandes Vaz et al., 2019; Forman, 2005; Maletzke et al., 2019; Moreira dos Reis et al., 2018; Moreo and Sebastiani, 2022; Pérez-Gállego et al., 2019, 2017; Schumacher et al., 2021)).

For the single-label multiclass case (which is the closest to our concerns) the APP needs to take a slightly different form, since the number of vectors $\mathbf{p} = (p(y_1), ..., p(y_n))$ representing valid distributions for arbitrary n is combinatorially high, for any reasonable grid of class prevalence values. As a solution, one can generate a number of random points on the probability simplex, without constraining the individual class prevalence values to lie on a predetermined grid; when this number is high enough, it probabilistically covers the entire spectrum of valid combinations.

However, even this form of the APP is not directly applicable to the multi-label scenario, because in this latter the class prevalence values in a valid vector do not necessarily sum up to 1. One could attempt to simply treat the multi-label problem as a set of independent binary problems, and then apply the

APP independently to each of the classes. Unfortunately, such a solution is impractical, for at least three reasons:

- The first reason is that the number of samples thus generated is exponential in n, since there are $m|\mathbf{g}|^n$ such combinations. Note that n (the number of classes in the codeframe) cannot be set at will since it is fixed, and thus, in order to keep the number of combinations tractable in cases in which n is large (in our experiments we use datasets with up to $n = 983$ classes), one would be compelled to set $m = 1$ and choose a very coarse grid $\mathbf{g}$ of values (this would anyway prove insufficient when dealing with large codeframes).
- The second and perhaps most problematic reason is that, in any case, many of the combinations are not even realisable. That is, there may be prevalence vectors for which no sample could be drawn at all. To see why, assume that, among others, we have classes y_1, y_2, y_3 in our codeframe, and assume that in our test set U, every time a datapoint is labelled with y_1 it is also labelled with either y_2 or y_3 but not both. This means that all samples σ for which prevalence values $p_\sigma(y_1) \neq (p_\sigma(y_2) + p_\sigma(y_3))$ are requested, cannot be generated.
- Yet another reason why applying the APP would be, in any case, undesirable, is that the classes in most multi-label datasets typically follow a power-law distribution, i.e., there are few very popular classes and a long tail of many rare, or extremely rare, classes. The APP will sometimes impose high prevalence values for classes that in reality are very rare, which means that the sampling must be carried out *with replacement*; this would generate samples consisting of many replicas of the same few datapoints, which is clearly undesirable.

For all these reasons we have designed a brand new protocol for MLQ, that we call ML-APP, since it is an adaptation of the APP to multi-label datasets. The protocol amounts to performing multiple rounds of the APP, each targeting a specific class, but with the range of prevalence values explored for each class being limited by the amount of available positive examples. This allows all samples to be drawn *without* replacement. In each round, a class y_i is actively sampled at controlled prevalence values while the prevalence values for the remaining classes are not predetermined.

The ML-APP covers the entire spectrum of class prevalence values, by drawing without replacement, for every single class. This means that, for large enough classes, there will be samples for which the prevalence of the class exhibits a large prior probability shift with respect to the training prevalence, while for rare classes the amount of shift will be limited by the availability of positive examples. Note that, when actively sampling a class y_i, any other class y_j will co-occur with it with a probability that depends on the correlation between y_i and y_j. For cases in which the class y_i being sampled is completely independent of the class y_j, the samples generated will display a class prevalence for y_j that is approximately similar to the prevalence of y_j in U. In other words, samples generated via the ML-APP have a desirable property, i.e., they preserve the stochastic correlations between the classes while also exhibiting widely varying

degrees of prior probability shift. Finally, note that the total number of samples that can be generated via the ML-APP can vary from dataset to dataset (even if they have the same number of classes), and depends on the actual number of positive instances for each class that are contained in the dataset. In any case, the maximum number of samples that can be generated via the ML-APP is bounded by $mn|\mathbf{g}|$.

4 Performing Multi-Label Quantification

In this section we present the multi-label quantification methods that we will experimentally compare in Section 5. Throughout this paper we will focus on *aggregative* quantification methods, i.e., methods that require all unlabelled datapoints to be classified (by a hard or a soft classifier, depending on the method) as an intermediate step, and that aggregate the individual (hard or soft) predictions in some way to generate the class prevalence estimates. The reason why we focus on aggregative methods is that they are by far the most popular quantification methods in the literature, and that this focus allows us an easier exposition. We will later show how the most interesting intuitions for performing MLQ that we discuss in this paper also apply to the non-aggregative case.

4.1 Multi-Label Quantification

In this paper we will describe and compare many different (aggregative) MLQ methods. In order to better assess their relative merits, we subdivide them into four different groups, depending on whether the correlations between different classes are exploited in the classification phase (i.e., by the classifier which provides input to an aggregative quantifier), or in the aggregation phase (i.e., in the phase in which the individual predictions are aggregated), or in both phases, or in neither of the two phases.

The first and simplest such group is that of MLQ methods that treat each class as completely independent, and thus solve n independent binary quantification problems. We call such an approach BC+BA ("binary classification followed by binary aggregation"), since in both the classification phase and the aggregation phase we treat the multi-label task as n independent binary tasks; we thus disregard, in both phases, the correlations among classes when predicting their class prevalence values. This is similar to the binary relevance (BR) problem transformation for classification, and consists of transforming the multi-label dataset L into a set of binary datasets $L_1, \ldots, L_n$ in which $L_i = \{(\mathbf{x}, \mathbf{1}[y_i \in Y]) : (\mathbf{x}, Y) \in L\}$ is labelled according to $\mathcal{Y}_i = \{\mathbf{0}, \mathbf{1}\}$, since the datapoints are relabelled using the indicator function $\mathbf{1}[z]$ that returns $\mathbf{1}$ (the minority class) if z is true or $\mathbf{0}$ (the majority class) otherwise. BC+BA methods then train one quantifier q_i for each training set L_i. At inference time, the prevalence vector for a given sample σ is computed as $\mathbf{p}_\sigma^{\text{BC+BA}} = (p_\sigma^{q_1}(\mathbf{1}), p_\sigma^{q_2}(\mathbf{1}), \ldots, p_\sigma^{q_n}(\mathbf{1}))$. Although this is technically a multi-label quantification method, BC+BA is actually the trivial solution that we expect any truly multi-label quantifier to beat.

A second, less trivial group is that of MLQ methods based on the use of binary aggregative quantifiers that receive input from (truly) multi-label classifiers. Methods in this group consist of n independent binary aggregative quantifiers that rely on the (hard or soft) predictions returned by a classifier natively designed to tackle the multi-label problem. Each binary quantifier takes into account only the predictions for its associated class, disregarding the predictions for the other classes. This represents a straightforward solution to the MLQ problem, as it simply combines already existing technologies (binary aggregative quantifiers built via off-the-shelf methods and (truly) multi-label classifiers built via off-the-shelf methods). In such a setting, the classification stage is influenced by the class-class correlations, but the quantification methods in charge of producing the class prevalence estimates for each class do not pay attention to any such correlation, and are disconnected from each other. Since methods in this group will consist of a (truly) multi-label classification phase followed by a binary quantification phase, we will refer to this group of methods as MLC+BA.

We next propose a third group of MLQ systems, i.e., ones consisting of natively multi-label quantification methods that receive ad input the outputs of n independent binary classifiers. Methods like these represent a non-trivial novel solution for the field of quantification, because no natively multi-label quantification method has been proposed so far in the literature; in Section 4.1.1 we propose some such methods. In order to clearly evaluate the merits of such a multi-label aggregation phase, as the underlying classifiers we use independent binary classifiers only. For this reason, we will call this group of methods BC+MLA.

The methods in the fourth and last group that we consider consist of combinations of a (truly) multi-label classification method and a (truly) multi-label quantification method among our newly proposed ones; this allows to exploit the class dependencies both at the classification stage and at the aggregation stage. We call this group of methods MLC+MLA.

Figure 1 illustrates in diagrammatic form the four types of multi-label quantification methods we study in this paper. In order to generate members of these four classes, we already have off-the-shelf components for implementing the binary classification, multi-label classification, and binary aggregation phases, but we have no known method from the literature to implement multi-label aggregation; Sections 4.1.1 and 4.1.2 are devoted to proposing two novel methods of this type.

4.1.1 Exploiting Class-Class Correlations at the Aggregation Stage by means of Regression Let us assume we have a multi-label quantifier q of type BC+BA or MLC+BA. Our idea is to detect how quantifier q fails in capturing the correlations between classes, and to correct q accordingly. This is somehow similar to the type of correction implemented in ACC (with respect to CC) and PACC (with respect to PCC). However, we will formalize this intuition as a general regression problem, thus not necessarily assuming this correction to be linear (as ACC and PACC instead do).

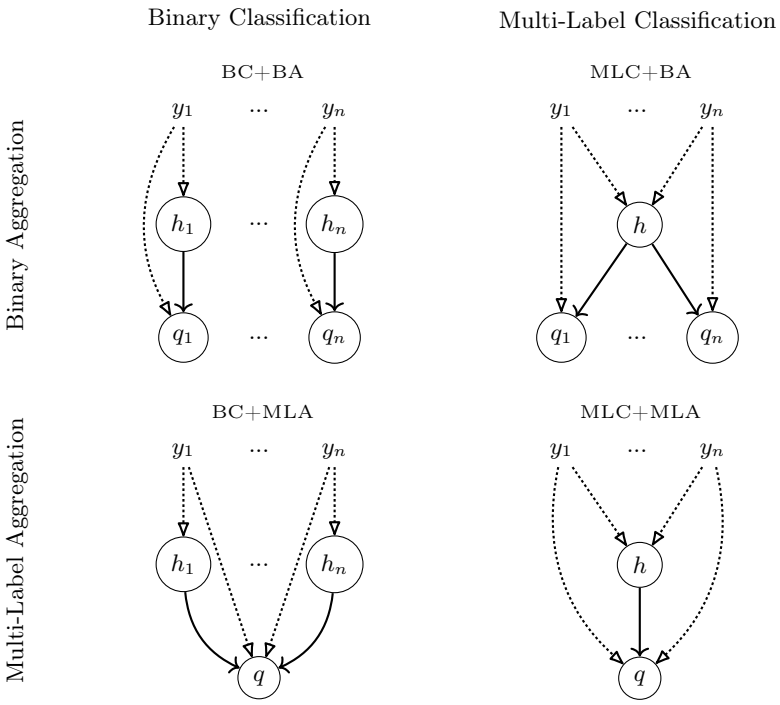

Fig. 1. The four groups of multi-label quantification methods. Dotted lines connecting class labels with a model (classifier or quantifier) indicate that the model learns from (or has access to) the class labels of the training datapoints. Solid lines connecting classifiers with quantifiers indicate a transfer of outputs from the classifier to the quantifier. With a slight deviation from our notation, here h denotes any classifier, hard or soft.

Roughly speaking, the idea that underlies our method is that of learning a regression function $r : \mathbb{R}^n \to \mathbb{R}^n$ that takes as input the vector of prevalence values as estimated by q, and returns a vector of *corrected* prevalence values. More concretely, we split our training set L into two parts, L_Q (that we use for training our quantifier q) and L_R (that we use for training a regressor r i.e., a function $r : \mathbb{R}^n \to \mathbb{R}^n$).[1] We then use the ML-APP protocol described in Section 3 to extract, from set L_R, a new training set $\mathcal{R} = \{\sigma_i \sim \text{ML-APP}(L_R, k, m, \mathbf{g})\}$ of l samples, where k (sample size), m (number of samples to draw for each prevalence value on the grid), and $\mathbf{g}$ (grid of prevalence values) are the parameters of the ML-APP protocol.

Having done this, we first train our quantifier q on L_Q. Note that, since q is a multi-label quantifier, it is a function that, given a sample σ, returns a vector $\hat{\mathbf{p}}^q_\sigma$ of n class prevalence values, not necessarily summing up to 1. We then

apply q to all the samples in our newly created dataset $\mathcal{R}$. As a result, for each sample $\sigma_i \in \mathcal{R}$, we haveobtain a pair $(\hat{\mathbf{p}}^q_{\sigma_i}, \mathbf{p}_{\sigma_i})$, where $\hat{\mathbf{p}}^q_{\sigma_i}$ is the vector of the n prevalence values estimated by q, and $\mathbf{p}_{\sigma_i}$ is the vector of the n true prevalence values. We use this set of l pairs as the training set for training a multi-output regressor $r : \mathbb{R}^n \to \mathbb{R}^n$ that takes as input a vector of n "uncorrected" prevalence values (i.e., values generated without exploiting the class-class correlations) and returns a vector of n "corrected" prevalence values (i.e., values generated by exploiting the class-class correlations); for training the regressor we can use any off-the-shelf multi-output regression algorithm. Note that the regressor indeed captures the correlations between classes, since it receives as input, for each sample, the class prevalence estimates for all the n classes.[2]

At inference time, given an (unlabelled) sample σ, we first obtain a preliminary estimate of the class prevalence values $\hat{\mathbf{p}}^q_\sigma$ by means of q, and then apply the correction learned by the regressor r, thus computing $\hat{\mathbf{p}}^r_\sigma = r(\hat{\mathbf{p}}^q_\sigma)$. We then normalize, by means of clipping,[3] every prevalence value in $\hat{\mathbf{p}}^r_\sigma$ so that it falls in the $[0, 1]$ interval, and return the estimate.

As noted above, the regressor exploits the class-class correlations during the aggregation phase. This means that, according to the subdivision of MLQ methods illustrated in Table 1, the addition of a regression layer on top of an existing quantifier q has the effect of transforming a BC+BA method into a BC+MLA method, or of transforming a MLC+BA method into a MLC+MLA method.

4.1.2 Exploiting Class-Class Correlations at the Aggregation Stage by means of Label Powersets We investigate an alternative way of modelling class-class correlations at the quantification level, this time by gaining inspiration from label powersets (LPs – see (Spolaôr et al., 2013)) and the heuristics for making their application tractable.

LP is a problem transformation technique devised for transforming any multi-label classification problem into a single-label one by replacing the original codeframe with another one that encodes subsets of this codeframe into "synthetic" classes. This problem transformation is directly applicable to the case of quantification as well. Of course, the combinatorial explosion of the number of synthetic classes has to be controlled somehow but, fortunately enough, the same heuristics investigated for MLC can come to the rescue.

Our method (which we here call LPQ, for "label powerset -based quantification") consists of generating, by means of any existing clustering algorithm, a set $\mathcal{C}$ of (non-overlapping) clusters consisting of few classes each, before applying the LP strategy, so that the number of possible synthetic classes remains under reasonable bounds. For example, if our codeframe has $n = 100$ classes, extracting 25 clusters of 4 classes each results in the maximum possible number of synthetic classes being $25 \cdot 2^4 = 400$, which is much smaller than the original 2^{100}. We perform this clustering by treating classes in $\mathcal{Y}$ as instances and training datapoints as features, so that a class is represented by a binary vector of datapoints, where 1 indicates that the datapoint belongs to the class and 0 that it does not. The clustering algorithm is thus expected to put classes displaying similar as-

signment patterns (i.e., classes that tend to label the same documents) in the same cluster.

Once we have performed the clustering, given the subset of classes $\mathcal{Y}_c \subseteq \mathcal{Y}$ contained in each cluster $c \in \mathcal{C}$, we need to convert the multi-label assignments into single-label assignments. This amounts to defining a mapping $2^{\mathcal{Y}_c} \rightarrow \mathcal{Y}'_c$, so that, e.g., the set of classes $\{y_1, y_5, y_6\} \subseteq \mathcal{Y}_c$ corresponds to a new synthetic class $y_{1:5:6} \in \mathcal{Y}'_c$. Once (single) labels have been assigned, we can train a single-label quantifier. This process is independently carried out for each cluster. codeframe take the single-label codeframe $\mathcal{Y}'_c$ determined from the $2^{\mathcal{Y}} \rightarrow \mathcal{Y}'$ multi-label-to-single-label mapping (a mapping that, e.g., would attribute to the set of classes $\{y_1, y_5, y_6\} \subseteq \mathcal{Y}_c$ the synthetic class $y_{1:5:6} \in \mathcal{Y}'_c$) and train a single-label quantifier on it; this needs to be repeated for each cluster. At inference time, in order to provide class prevalence estimates for the classes in $\mathcal{Y}_c$ from the predictions made for the classes in $\mathcal{Y}'_c$ by the above-mentioned quantifier, we have to "reverse" the multi-label-to-single-label mappingassignment. This process is straightforward since the mapping is bijective. By doing so, we can reconstruct the estimated prevalence value for class $y_i \in \mathcal{Y}_c$ as , so that the estimated prevalence value of $y_i \in \mathcal{Y}_c$ is the sum of the estimated prevalence values of all labels $y' \in \mathcal{Y}'_c$ that involve y_i.; performing this for each cluster $c \in \mathcal{C}$ returns prevalence estimates for all classes $y_i \in \mathcal{Y}$. This process is repeated for each cluster $c \in \mathcal{C}$ in order to obtain prevalence estimates for all classes $y_i \in \mathcal{Y}$.

More formally, let us define a matrix $\mathbf{A}$ that records the label assignment in cluster c, so that $a_{ij} = 1$ if the set of classes represented by the synthetic class $y'_i \in \mathcal{Y}'_c$ contains class $y_j \in \mathcal{Y}_c$, and $a_{ij} = 0$ if this is not the case. Note that $\mathbf{A}$ has as many rows as there are classes in $\mathcal{Y}'_c$ and as many columns as there are classes in $\mathcal{Y}_c$. Once our single-label quantifier q produces an output $\hat{\mathbf{p}}^q_\sigma$, we only need to compute the product $(\hat{\mathbf{p}}^q_\sigma)^\top \mathbf{A}$ to obtain the vector of prevalence estimates for the classes in $\mathcal{Y}_c$. Performing all this for each cluster $c \in \mathcal{C}$ returns prevalence estimates for all classes $y_i \in \mathcal{Y}$.

In principle, the disadvantage of this method is that it cannot learn the correlations between classes that belong to different clusters. However, the method is based on the intuition that classes that are indeed correlated tend to end up in the same cluster, and that the inability to model correlations between classes that belong to different clusters will be more than compensated by the reduction in the number of combinations that one needs to take into account.

5 Experiments

In this section we turn to describing the experiments we have carried out in order to evaluate the performance of the different methods for MLQ that we have presented in the previous sections. In Section 5.1 we discuss the evaluation measure we adopt, while in Section 5.2 we describe the datasets on which we perform our experiments. The results, omitted here for reasons of space, can be found in the extended version of this paper at `https://dl.acm.org/doi/10.1145/3606264`.

5.1 Evaluation Measures

Any evaluation measure for binary quantification can be easily turned into an evaluation measure for multi-label quantification, since evaluating a multi-label quantifier can be done by evaluating how well the prevalence value $p(y_i)$ of each class $y_i \in |\mathcal{Y}|$ is approximated by the prediction $\hat{p}(y_i)$. As a result, it is natural to take a standard measure $d(\mathbf{p}, \hat{\mathbf{p}})$ for the evaluation of binary quantification, and turn it into a measure

$$\mathrm{D}(\mathbf{p}, \hat{\mathbf{p}}) = \frac{1}{n} \sum_{i=1}^{n} d((p_i, (1-p_i)), (\hat{p}_i, (1-\hat{p}_i))) \tag{1}$$

for the evaluation of multi-label quantification. (This is exactly what we do in multi-label *classification*, in which we take F_1, a standard measure for the evaluation of binary classification, and turn it into macroaveraged F_1, which is the standard measure for the evaluation of multi-label classification.)

The study of evaluation measures for binary (and single-label multiclass) quantification performed in (Sebastiani, 2020) concludes that the most satisfactory such measures are *absolute error* and *relative absolute error*; these are the two measures that we are going to use in this paper. In the binary case, absolute error is defined as

$$\begin{aligned} \mathrm{ae}(\mathbf{p}, \hat{\mathbf{p}}) &= \frac{|p_1 - \hat{p}_1| + |p_2 - \hat{p}_2|}{2} \\ &= \frac{|p_1 - \hat{p}_1| + |(1-p_1) - (1-\hat{p}_1)|}{2} \\ &= |p_1 - \hat{p}_1| \end{aligned} \tag{2}$$

which yields the multi-label version

$$\mathrm{AE}(\mathbf{p}, \hat{\mathbf{p}}) = \frac{1}{n} \sum_{i=1}^{n} |p_i - \hat{p}_i| \tag{3}$$

In the binary case, relative absolute error is instead defined as

$$\begin{aligned} \mathrm{rae}(\mathbf{p}, \hat{\mathbf{p}}) &= \frac{1}{2} \left(\frac{|p_1 - \hat{p}_1|}{p_1} + \frac{|p_2 - \hat{p}_2|}{p_2} \right) \\ &= \frac{1}{2} \left(\frac{|p_1 - \hat{p}_1|}{p_1} + \frac{|(1-p_1) - (1-\hat{p}_1)|}{(1-p_1)} \right) \end{aligned} \tag{4}$$

which yields the multi-label version

$$\mathrm{RAE}(\mathbf{p}, \hat{\mathbf{p}}) = \frac{1}{2n} \sum_{i=1}^{n} \left(\frac{|p_i - \hat{p}_i|}{p_i} + \frac{|(1-p_i) - (1-\hat{p}_i)|}{(1-p_i)} \right) \tag{5}$$

Since RAE is undefined when $p_i = 0$ or $p_i = 1$, we smooth the probability distributions $\mathbf{p}$ and $\hat{\mathbf{p}}$ via additive smoothing; in the binary case, this maps a distribution $\mathbf{p} = (p_i, (1-p_i))$ into

$$\mathrm{s}(\mathbf{p}) = \left(\frac{\epsilon + p_i}{2\epsilon + 1}, \frac{\epsilon + (1-p_i)}{2\epsilon + 1} \right) \tag{6}$$

with ϵ the smoothing factor; following (Forman, 2008), we set $\epsilon = (2|\sigma|)^{-1}$.

In the experiments we describe in Section 5, the trends we observe and the conclusions we draw for AE hold for RAE as well. In Section 5 we will thus report our results in terms of AE only, deferring the results in terms of RAE to (Moreo et al., 2024).

5.2 Datasets

For our experiments we use 15 popular MLC datasets, including 3 datasets specific to text classification (Reuters-21578,[4] Ohsumed (Hersh et al., 1994), and RCV1-v2[5]), plus all the datasets linked from the SCIKIT-MULTILEARN package (Szymanski and Kajdanowicz, 2017) with the exception of the RCV1-v2 subsets (we omit them since we already include the much larger collection from which they were extracted). We refer to the original sources for detailed descriptions of these datasets.[6]

For the three textual datasets, we apply lowercasing, stop word removal, and punctuation removal, as implemented in SCIKIT-LEARN,[7] and mask numbers with a special token. We retain all terms appearing at least 5 times in the training set, and convert the resulting set of words into (sparse) tfidf-weighted vectors using SCIKIT-LEARN's default vectorizer.[8]

For all datasets, we remove very rare classes (i.e., those with fewer than 5 training examples) from consideration, since they pose a problem when it comes to generating validation (i.e., held-out data) sets. Indeed, since we optimize the hyperparameters for all the methods we use (as explained below), we need validation sets, and it is sometimes impossible to have positive examples for these classes in both the training and validation sets (let us remember that pure stratification in multi-label datasets is not always achievable, as argued in (Sechidis et al., 2011; Szymański and Kajdanowicz, 2017)). Note that all this only concerns the training set, and has nothing to do with the test set, which can include (and indeed includes, for most datasets) extremely rare classes, since removing classes that are rare in the test set would lead to an unrealistic experimentation. Note also that removing classes that are rare in the training set is "fair", i.e., equally affects all methods that we experimentally compare, since all of them involve hyperparameter optimization. Finally, note that, whenever a method requires generating *additional* (and maybe nested) validation sets, it is inevitably exposed to the problems mentioned above, and can thus be at a disadvantage with respect to other methods that do not require additional validation data. (Moreo et al., 2024) gives a complete description of the datasets we use (after deleting rare classes), along with some useful statistics proposed in (Read, 2010; Zhang and Zhou, 2014), and shows the distribution of prevalence values for each dataset. Note that, in most datasets, this distribution obeys a power law.

We set the parameters of the ML-APP for generating test samples (see Section 3) as follows. We fix the sample size to $k = 100$ in all cases. We set the grid of prevalence values to $\mathbf{g} = \{0.00, 0.01, \ldots, 0.99, 1.00\}$ in all cases but for dataset `Delicious`, since in this latter the number of combinations thus generated would be intractable, given that this is dataset with no fewer than 983 classes; for

`Delicious` we use the coarser-grained grid $\mathbf{g} = \{0.00, 0.05, \ldots, 0.95, 1.00\}$. We set m (the number of samples to be drawn for each prevalence value) independently for each dataset, to the smallest number that yields more than 10,000 test samples (m ranges from 1 in `Delicious` to 40 in `Birds`).

We break down the results into three groups, each corresponding to a different amount of shift. The rationale behind this choice is to allow for a more meaningful analysis of the quantifiers' performance, since the APP (and, by extension, the ML-APP) has often been the subject of criticism for generating samples exhibiting degrees of shift that are judged unrealistic and unlikely to occur in real cases (Esuli and Sebastiani, 2015; Hassan et al., 2021). We instead believe that general-purpose quantification methods should be tested in widely varying situations, from low-shift to high-shift ones, and we thus prefer to test all such scenarios, but split the corresponding results into groups characterized by of more or less homogeneous amounts of shift.

More specifically, for each test sample generated via the ML-APP, we compute its prior probability shift with respect to the training set in terms of AE between the vectors of training and test class prevalence values. We then bring together all the resulting shift values and split the range of such values in three equally-sized intervals (that we dub *low shift*, *mid shift*, and *high shift*). The accuracy values we report are thus not averages across the same number of experiments, since the ML-APP often tends to generate more samples in the low-shift region than samples in the mid-shift region and (above all) in the high-shift region. The number of samples, as well as the distribution of shift values, depends on each dataset.

The results of our experiments are omitted here for reasons of space, and can be found in the extended version of this paper at `https://dl.acm.org/doi/10.1145/3606264`. The results clearly show that there is an ordering BC+BA $\prec$ MLC+BA $\prec$ BC+MLA $\prec$ MLC+MLA, in which $\prec$ means "performs worse than", which holds, independently of the base quantifier of choice, in almost all cases. The same experiments also indicate that *there is a substantial improvement in performance that derives from simply replacing the binary classifiers with one multi-label classifier* (moving from BC+BA to MLC+BA or from BC+MLA to MLC+MLA), i.e., from bringing to bear the class-class correlations at the classification stage, and that *there is an equally substantial improvement when binary aggregation is replaced by multi-label aggregation* (switching from BC+BA to BC+MLA or from MLC+BA to MLC+MLA), i.e., when the class-class correlations are exploited at the aggregation stage. What also emerges from these results is that, consistently with the above observations, *the best-performing group of methods is* MLC+MLA, i.e., methods that explicitly take class dependencies into account *both* at the classification stage and at the aggregation stage. Methods that learn from the stochastic correlations among the classes perform way better than methods that do not, even in the low-shift regime. Overall, the best-performing method on average is MLC+MLA when equipped with PCC as the base quantifier.

6 Conclusions

In this paper we have investigated MLQ, a quantification task which had remained, since the origins of quantification research, essentially unexplored.

The first contribution of this paper is ML-APP, the first protocol for the evaluation of MLQ systems that is able to confront these systems with samples that exhibit from low to high levels of prior probability shift while at the same time preserving the stochastic correlations between the classes.

As a second contribution, we have also described and experimentally compared a number of MLQ methods. For ease of exposition, we have particularly focused on multi-label quantifiers that work by aggregating predictions for individual datapoints issued by a classifier ("aggregative" multi-label quantifiers), and have subdivided them into four groups, based on whether the correlations between classes are brought to bear in the classification stage (MLC+BA), in the quantification stage (BC+MLA), in both stages (MLC+MLA), or in neither of the two stages (BC+BA). Some of these methods (specifically: those in the BC+BA and MLC+BA groups) are trivial combinations of available classification and quantification methods, while others (specifically: those in the BC+MLA and MLC+MLA groups) are non-obvious, and proposed here for the first time. The thorough experimentation (reported in (Moreo et al., 2024)) that we have carried out on a large number of datasets has clearly shown that there is a substantial improvement in performance that derives from simply replacing binary classifiers with truly multi-label classifiers (i.e., from switching from BC to MLC), and that there is an equally substantial improvement when binary aggregation is replaced by truly multi-label aggregation (i.e., when switching from BA to MLA). Consistently with these two intuitions, MLC+MLA methods unequivocally prove the best of the lot; of the two MLC+MLA methods we have proposed, RQ proves clearly superior to LPQ. In the light of this superiority of MLA with respect to BA, it is also interesting that both RQ and LPQ can be straightforwardly used in association to non-aggregative quantifiers too.

Acknowledgments

The work of A. Moreo and F. Sebastiani has been supported by the SoBigData++ project, funded by the European Commission (Grant 871042) under the H2020 Programme INFRAIA-2019-1, by the AI4Media project, funded by the European Commission (Grant 951911) under the H2020 Programme ICT-48-2020, and by the SoBigData.it and FAIR projects funded by the Italian Ministry of University and Research under the NextGenerationEU program; the authors' opinions do not necessarily reflect those of the funding agencies. The work of M. Francisco has been supported by the FPI 2017 predoctoral programme, from the Spanish Ministry of Economy and Competitiveness (MINECO), grant BES-2017-081202.

Bibliography

Bunse, M., Moreo, A., Sebastiani, F., and Senz, M. (2022). Ordinal quantification through regularization. In *Proceedings of the 33rd European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML / PKDD 2022)*, Grenoble, FR. Forthcoming.

Card, D. and Smith, N. A. (2018). The importance of calibration for estimating proportions from annotations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2018)*, pages 1636–1646, New Orleans, US.

Esuli, A., Moreo, A., and Sebastiani, F. (2018). A recurrent neural network for sentiment quantification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM 2018)*, pages 1775–1778, Torino, IT.

Esuli, A., Moreo, A., and Sebastiani, F. (2022). LeQua@CLEF2022: Learning to Quantify. In *Proceedings of the 44th European Conference on Information Retrieval (ECIR 2022)*, pages 374–381, Stavanger, NO.

Esuli, A. and Sebastiani, F. (2015). Optimizing text quantifiers for multivariate loss functions. *ACM Transactions on Knowledge Discovery and Data*, 9(4):Article 27.

Fernandes Vaz, A., Izbicki, R., and Bassi Stern, R. (2019). Quantification under prior probability shift: The ratio estimator and its extensions. *Journal of Machine Learning Research*, 20:79:1–79:33.

Forman, G. (2005). Counting positives accurately despite inaccurate classification. In *Proceedings of the 16th European Conference on Machine Learning (ECML 2005)*, pages 564–575, Porto, PT.

Forman, G. (2008). Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, 17(2):164–206.

Hassan, W., Maletzke, A. G., and Batista, G. (2021). Pitfalls in quantification assessment. In *Proceedings of the CIKM 2021 Workshop on Learning to Quantify*, Virtual Event.

Hersh, W., Buckley, C., Leone, T., and Hickman, D. (1994). OHSUMED: An interactive retrieval evaluation and new large text collection for research. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1994)*, pages 192–201, Dublin, IE.

Maletzke, A., Moreira dos Reis, D., Cherman, E., and Batista, G. (2019). DyS: A framework for mixture models in quantification. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, pages 4552–4560, Honolulu, US.

Moreira dos Reis, D., Maletzke, A. G., Silva, D. F., and Batista, G. E. (2018). Classifying and counting with recurrent contexts. In *Proceedings of the 24th ACM International Conference on Knowledge Discovery and Data Mining (KDD 2018)*, pages 1983–1992, London, UK.

Moreo, A., Francisco, M., and Sebastiani, F. (2024). Multi-label quantification. *ACM Transactions on Knowledge Discovery and Data*, 18(1):Article 4.

Moreo, A. and Sebastiani, F. (2022). Tweet sentiment quantification: An experimental re-evaluation. *PLOS ONE*, 17(9):1–23.

Pérez-Gállego, P., Castaño, A., Quevedo, J. R., and del Coz, J. J. (2019). Dynamic ensemble selection for quantification tasks. *Information Fusion*, 45:1–15.

Pérez-Gállego, P., Quevedo, J. R., and del Coz, J. J. (2017). Using ensembles for problems with characterizable changes in data distribution: A case study on quantification. *Information Fusion*, 34:87–100.

Read, J. (2010). *Scalable multi-label classification*. PhD thesis, University of Waikato, Hamilton, NZ.

Schumacher, T., Strohmaier, M., and Lemmerich, F. (2021). A comparative evaluation of quantification methods. arXiv:2103.03223.

Sebastiani, F. (2020). Evaluation measures for quantification: An axiomatic approach. *Information Retrieval Journal*, 23(3):255–288.

Sechidis, K., Tsoumakas, G., and Vlahavas, I. (2011). On the stratification of multi-label data. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD 2011)*, pages 145–158, Athens, GR.

Spolaôr, N., Cherman, E. A., Monard, M. C., and Lee, H. D. (2013). A comparison of multi-label feature selection methods using the problem transformation approach. *Electronic Notes in Theoretical Computer Science*, 292:135–151.

Szymanski, P. and Kajdanowicz, T. (2017). A scikit-based Python environment for performing multi-label classification. arXiv:1702.01460 [cs.LG].

Szymański, P. and Kajdanowicz, T. (2017). A network perspective on stratification of multi-label data. In *Proceedings of the 1st International Workshop on Learning with Imbalanced Domains: Theory and Applications (LIDTA 2017)*, pages 22–35, Skopje, MK.

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