The Importance of the Test Set Size in Quantification Assessment

André Maletzke[1] , **Waqar Hassan**[2] , **Denis dos Reis**[2] and **Gustavo Batista**[3]
[1]Western Paraná State University, Foz do Iguaçu, Brazil
[2]University of São Paulo, São Carlos, Brazil
[3]University of New South Wales, Sydney, Australia
andre.maletzke@unioeste.br, {waqar, denismr}@usp.br, g.batista@unsw.edu.au

Abstract

Quantification is a task similar to classification in the sense that it learns from a labeled training set. However, quantification is not interested in predicting the class of each observation, but rather measure the class distribution in the test set. The community has developed performance measures and experimental setups tailored to quantification tasks. Nonetheless, we argue that a critical variable, the size of the test sets, remains ignored. Such disregard has three main detrimental effects. First, it implicitly assumes that quantifiers will perform equally well for different test set sizes. Second, it increases the risk of cherry-picking by selecting a test set size for which a particular proposal performs best. Finally, it disregards the importance of designing methods that are suitable for different test set sizes. We discuss these issues with the support of one of the broadest experimental evaluations ever performed, with three main outcomes. (*i*) We empirically demonstrate the importance of the test set size to assess quantifiers. (*ii*) We show that current quantifiers generally have a mediocre performance on the smallest test sets. (*iii*) We propose a meta-learning scheme to select the best quantifier based on the test size that can outperform the best single quantification method.

1 Introduction

Quantification is a recently proposed Data Mining task to estimate the class distribution in a test set. This task has found numerous applications where the primary interest is *not* in the classification of individuals, but rather the understanding of the behavior of groups. Typical uses of quantification are, for instance, sentiment analysis when we want to estimate the proportion of positive tweets about a particular product [González *et al.*, 2017b]; quantify the marine ecology from images [Beijbom *et al.*, 2015]; or, mosquito surveillance when we need to calculate the number of disease-carrying mosquitoes captured by an insect trap [Chen *et al.*, 2014; Silva *et al.*, 2015; Maletzke *et al.*, 2018b].

Quantification and classification are closely related areas. The simplest quantification method, classify and count (CC), merely uses the outcome of a classifier to count how many objects belong to each class. Nevertheless, classification and quantification have fundamental differences, the main one being that classifiers provide predictions for individual data points, whereas quantifiers issue predictions for sets with several instances.

Therefore, we should consider such differences when designing and evaluating quantification methods. In particular, the size of the test set is a characteristic that can affect the performance of quantification methods. In classification, the relationship between the training set size and error rate is a recurring theme of investigation of practitioners and theorists, being the focal point of the statistical learning theory [Vapnik, 2013]. As a consequence, quantification methods that adopt classifiers in intermediate steps naturally inherits this relationship. Therefore, they can benefit from the same theoretical and practical findings in literature. Conversely, the size of the test set is usually not a point of concern for classification. However, the size of the test set influences the quality of a quantifier and must be regarded in the design, analysis and evaluation of new quantification methods.

Thus far, the quantification community has mostly ignored the importance of test set sizes. Unfortunately, such disregard has detrimental effects on the quality of the research in the area. First, it implicitly assumes that a method that performs well for large test sets will perform equally well for small test sets. Such an assumption can be invalidated by a simple experiment as the one shown in Figure 1. In this figure, we summarize a comparison between five well-known quantifiers on several benchmark datasets with test set sizes of 500 instances (*A*) and ten instances (*B*). The ranking of the algorithms has significantly changed depending on the size of the test set, even though all other variables remained unchanged.

Second, not explicitly acknowledging the effects of varying test sample size increases the chances of cherry picking in quantification research. For instance, a paper that proposes a new quantification method that relies on estimating several statistics from the test set may deliberately choose to evaluate the proposal only on large test sets, where the method benefits the most: with more data, there is a higher chance of accurately estimating the required statistics, making the method more competitive.

Finally, by ignoring the importance of the test sample size, we disregard this variable in the design of quantification al-

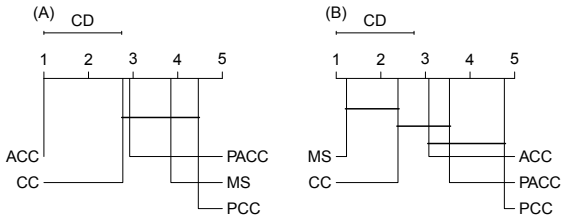

Figure 1: Nemenyi's post hoc test for mean absolute quantification error. Groups of methods which are not significantly different at $p < 0.05$ are connected.

gorithms. As different applications may pose distinct requirements, the community should be focusing on proposing methods that perform well on a variety of test sets sizes. From our experiments, we observe that several approaches perform well for larger test sets with more than 100 instances. On the other hand, for tiny test sets of 10 data points, only two strategies (Median Sweep and SORD) outperformed the arguably naive and frequently criticized Classify and Count algorithm. In our opinion, this is evidence that the community has given little attention to this setting, and further research needs to be developed to propose accurate methods for small test sets.

Although the smallest test sets are the most challenging to provide accurate results for, we do not think the community should entirely focus on them. We believe that most applications require quantifying sets of varying sizes. For instance, for the sentiment analysis and insect counting applications mentioned before, an important outcome is the number of positive tweets/captured insects over different periods of time, such as the last hour, day, and week. This setting leads to the operation of quantification methods on different test set sizes. We explore this idea by proposing a simple meta-learning strategy that selects the quantification algorithm according to the size of the test set at hand.

Our objective is to call the community attention to the importance of the test set size when designing and evaluating quantification methods. To support our claims, we define a new experimental design to evaluate quantification algorithms. To the best of our knowledge, we perform the most comprehensible evaluation of quantification methods in literature. We compare 12 quantification methods representative of different counting paradigms on 13 benchmark datasets, with 14 test set sizes and 100 class distributions.

Our main contributions are: (*i*) we empirically demonstrate the importance of the test set size to assess the performance of quantification methods; (*ii*) we show that current quantification methods generally have a mediocre performance on the smallest test sets; and (*iii*) we propose a meta-learning scheme to select the best quantifier based on the test set size that can outperform the best single quantification method.

At last, we advocate that, for now on, quantification research should use the experimental setup proposed in this paper to evaluate their proposals and compare to the state-of-the-art. To support these comparisons, we created a paper website[1] with all data, code and detailed results so that other researchers can directly use our results when evaluating newly-proposed methods.

This paper is organized as follows: Section 2 provides the necessary background information on the quantification task; Section 3 summarizes the relevant literature; Section 4 describes the experimental setup proposed in this paper; Section 5 presents our meta-learning framework for recommending quantifiers according to each test set; Section 6 discusses the experimental setup and Section 7 presents the empirical results and discussion of a comprehensible evaluation of quantification methods; Finally, Section 8 concludes this work and present directions for future work.

2 Background

Consider the labeled set $D = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\}$, in which each example $\vec{x}_i \in \mathcal{X}$ is a vector in the m-dimensional feature space $\mathcal{X}$, and $y_i \in \mathcal{C} = \{c_1, \ldots, c_k\}$ is its respective class label. We can formally define a quantifier as a model that predicts the prevalence of each class in a sample according to the following equation:

$$\mathcal{Q} : \mathbb{S}^{\mathcal{X}} \longrightarrow [0, 1]^k$$

where $\mathbb{S}^{\mathcal{X}}$ denotes the universe of possible samples from $\mathcal{X}$. Given a specific sample $S \in \mathbb{S}^{\mathcal{X}}$, the quantifier outputs a vector, $\hat{\vec{q}}(S) = [\hat{p}_1(S), \ldots, \hat{p}_k(S)]$, that estimates the prior probability for each respective class $c_1, \ldots, c_k$, subject to the constraint $\sum_{i=1}^{k} \hat{p}_i(S) = 1$. A good quantifier is one that produces a $\hat{\vec{q}}(S)$ that reasonably approximates the true class ratios $\vec{q} = [p_1(S), \ldots, p_k(S)]$ of the probability distribution from which S was sampled.

All interesting quantification problems undergo a class distribution drift from training to test data. Otherwise, we could use the class ratios from the training set to predict the distribution of the test sets. For our previous examples, sentiment analysis and insect counting, the class distribution in the training set is not a suitable predictor of the class ratio in newly observed data. More importantly, we expect the class distribution to change on multiple occasions over time. On sentiment analysis, the ratio of positive comments can vary with the development of events that affect public opinion or even due to the work of automated trolls commanded by a bad actor. In the insect quantification problem, both relative and absolute numbers of captured insects for each species vary according to the local population, circadian rhythm and ambient factors such as temperature [Maletzke *et al.*, 2018a].

3 Related Work

In the last decade, several algorithms for quantification have been proposed. Although they share the same general setting and objective, their introduction by different communities led to different names for the quantification task: *prevalence estimation* [Barranquero *et al.*, 2013], *class prior estimation* [Chan and Ng, 2006], and *class distribution estimation* [González-Castro *et al.*, 2013].

González et al. [2017a] organized most of the prior work according to similarities between algorithms, resulting in a

[1] sites.google.com/site/andregustavom/research/mlq-framework

taxonomy of quantification methods described by three different groups:

I. ***Classify, Count, and Correct*:** methods that classify each instance firstly and then count how many belong to each class. Methods that apply any correction to their predictions are included in this group as well;

II. ***Adapting traditional classification algorithms*:** algorithms that modify the mechanics of traditional classification learning methods so that they become quantifiers;

III. ***Distribution matching*:** algorithms that parametrically model the training distribution and later search the parameters that produce the best match against the test set.

Experiments in quantification are usually more involved than those in classification. The evaluation process is data hungry since it requires measuring the performance on entire samples with several data points. Moreover, we need to repeat the experiment to obtain confidence intervals, collect the test samples from a larger pool to guarantee some diversity among the samples, and vary the class distribution to simulate the expected data distribution variability.

Due to the similarity between the quantification and classification tasks, quantification experiments are typically conducted using classification datasets. The experimental setup usually follows these two steps: (*i*) each dataset is divided into two subsets using stratified sampling, resulting in training and test sets; and (*ii*) given the test set with n_{Te} data points, a collection of samples with length $m_{Te} \ll n_{Te}$ is extracted varying the class distribution randomly or according to a predefined range.

This approach allows simulating a range of class distribution drifts, and has been used by several authors, including Forman et al. [2005], Bella et al. [2010], Barranquero et al. [2013], Milli et al. [2013], Reis et al. [2018a] and Pérez-Gállego et al., [2019], and Maletzke et al. [2019].

However, to the best of our knowledge, every paper in quantification research uses a fixed and arbitrary m_{Te}. The only exception is the paper of Maletzke et al. [2019]. In that paper, the authors are interested in characterizing the performance of a smaller set of quantification methods and understand the importance of their internal parameters.

4 Designing experiments in quantification

The experimental design used in the vast majority of quantification papers ignore the influence of the test set size. In this paper, we argue this is a severe flaw since the performance of quantifiers fluctuates according to this variable. Identifying the performance in a single test sample size does not provide enough information to characterize the behavior of a method. For this reason, we include an additional step into the standard evaluation procedure so that we can assess the performance on distinct test sizes, in addition to varied class distributions. Algorithm 1 describes the steps to generate samples for the evaluation.

In Algorithm 1, $T_e^{c_i} \subset T_e$ includes all observations from T_e that belong to class c_i. The function $getSample$ returns a sample, given the number of observations and class distribution, using the examples from the T_e. For most datasets, due to limited data, different samples can share observations. However, one observation should not appear more than once in a same sample.

Algorithm 1: Generating test samples

Input: Test set T_e
Output: Set of test samples S_{T_e}
$S_s \leftarrow \{s_{min}, \ldots, \min(L, |T_e^{c_1}|, \ldots, |T_e^{c_k}|)\}$ {Sample sizes, $s_{min} > 1$ and L is a hard limit}
$Pr_{class} \leftarrow \{[p_1^1, ..., p_k^1], \ldots, [p_1^N, ..., p_k^N]\}$ {N class distributions}
$R_{sample} \leftarrow R$ {Replication}
for $i \leftarrow 1$, length(S_s) **do**
  for $j \leftarrow 1$, length(Pr_{class}) **do**
    for $k \leftarrow 1$, R_{sample} **do**
      $S_{T_e} \leftarrow S_{T_e} \cup \{\text{getSample}(T_e, S_s[i], Pr_{class}[j])\}$
    end for
  end for
end for
return S_{T_e};

5 Meta-learning Quantification

We argue that the best quantification algorithm for large sets may not be the best method for small sets, and vice versa. We later empirically confirm this hypothesis. Thus, two strategies could be applied to improve our evaluation procedure. First, the proposal of new quantifiers that can perform well independently of the size of the test set. Second, a mechanism to autonomously select the quantifier, from a group of options, that is the most adequate for a given test set. In this paper, we adopt the latter strategy: our proposal is not a new quantification algorithm by itself, but rather a framework that recommends quantifiers according to each test set. To achieve this effect, we employ an algorithm recommendation method with meta-learning. As stated by Pavel et al., [2009], meta-learning systems are used to indicate which algorithm should be used to reach the best possible results for each individual task, according to its particularities.

Our framework, named Meta-Learning Quantification (MLQ), applies meta-learning concepts to describe each problem and learn a recommender that predicts which quantifier should be used for each test set. Figure 2 shows the architecture of our framework.

First, we gather several datasets and split each one into two halves (training and test halves). The training half is used to learn all quantifiers that can be recommended for quantifying sets, which are extracted from the testing half. We also use the training half to learn our recommender, following path two in Figure 2. Then, we break up the training half into two new halves, which are the training and validation sets for the meta-learning procedure. From the validation half, we extract several test sets of different sizes and class distributions. For each one, we extract metafeatures and measure the performance of all quantifiers. As a result, a tuple composed of metafeatures and the identification of the best quantifier is recorded in a metatable (item four in Figure 2). We repeat this process several times to get a vast diversity of tuples. Af-

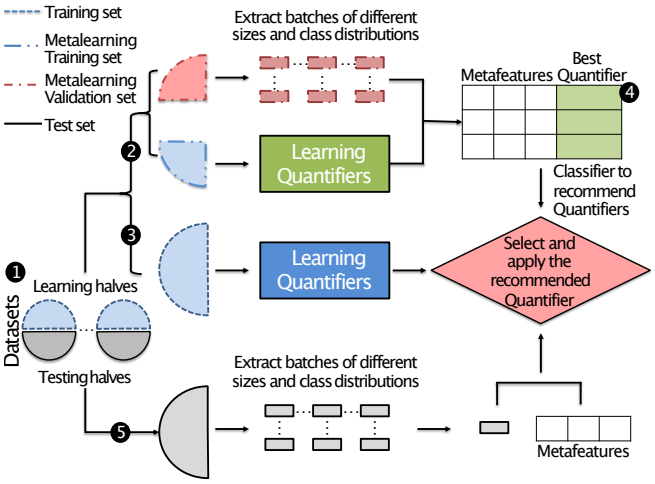

Figure 2: Framework architecture for quantifiers recommendation.

ter that, this metatable is used to induce our recommendation model, which is a classifier where the classes represent different quantification algorithms.

Meta-learning literature reports several metafeatures to describe characteristics of a dataset or a group of data points, such as information theory and statistics measures [Lemke *et al.*, 2015]. In our framework, in addition to the test size itself, as a numerical feature, we use simple statistic measures and the distance reported by the DyS framework when quantifying [Maletzke *et al.*, 2019]. The statistics measures used were: coefficient of variation, kurtosis, skewness, and correlation. The average value of all attributes was calculated to describe a set of observation points.

Finally, we follow the path three (Figure 2) to build the final quantifiers that can be chosen by the recommender in the test phase. Path five represents the experimental evaluation, described in the next section.

6 Experimental Setup

In this section, we describe a set of experiments with several well-known quantification algorithms, aiming to show the influence of varying test set size on their performance. Table 1 presents all binary quantifiers evaluated in our experiments. Methods such as CC, ACC, T50, X, MAX, and MS were proposed by Forman [2005; 2006]. The PACC method or also named Scaled Probability Average (SPA) was proposed by Bella et al., [2010]. Quantification trees were proposed by Milli et al., [2013] and EMQ by Saerens et al., [2002]. Finally, both distribution matching methods, HDy and DyS, were proposed by Gonzáles-Castro et al., [2013] and Maletzke et al., [2019], respectively.

DyS framework was evaluated including the top five similarity functions according to Maletzke et al. [2019]: ORD, Sample ORD (SORD), Probabilistic Symmetric (PS), Jensen Difference (JD), and Topsøe (TS). We report the positive class ratio that is the median for a varying number of bins, from 2 to 20 with increments of 2, following the findings.

We must note that, we only used binary quantification datasets. This allowed us to test a wider range of algorithms

Quantifier	Acronym	Taxonomy Group
Classify and Count	CC	
Adjust Classify and Count	ACC	
Probabilistic Classify	PCC	
Probabilistic Adjust Classify and Count	PACC	
Threshold Selection Method		
Set the decision threshold where $tpr = 50\%$	T50	I
Set the decision threshold where $(1 - tpr) = fpr$	X	
Set the decision threshold where $tpr - fpr$ is maximized	MAX	
Estimate *tpr* and *fpr* for several thresholds, returning the median of them	MS	
Quantification Trees	QT	II
Expectation Maximization Quantification	EMQ	
Distribution matching with Hellinger Distance	HDy	III
Mixture Model Framework	DyS	

Table 1: Quantifiers evaluated.

and simplify the design of experimental evaluation.

However, limiting our setup to binary quantification does not disqualify the point we are making regarding the relevance of test sample size. Indeed, if this variable affects binary quantification, it stands to reason that it also affects multiclass quantification.

We uniformly split each dataset into two parts: training and test. With the training part, we performed 10-fold cross-validation to obtain the classification scores used by the mixture models that follow the DyS framework, as well as to calculate the *tpr* and *fpr* performance statistics used by ACC and its variations. The full training half was also used to train a single scorer that was applied on the individual data points in the test half, to obtain a set of test scores. We produced all scores using Random Forests with 200 trees. Test samples were obtained from the test part according to Algorithm 1. We also use Random Forests to induce the recommendation model for MLQ framework.

We varied the test set size from 10 to 100 with increments of 10 data points, and from 100 to 500 with increases of 100. For each test set size, we varied the positive class proportion from 0% to 100% with increments of 1%. We execute 10 runs for each pair of test set size and class distribution.

As the time complexity of SORD grows linearithmic with the total number of observations, we undersampled training scores to $1,000$ per class, whenever there were more than that. EMQ has used five iterations as a stopping condition.

We measured the performance of quantifiers in our experiments using the Mean Absolute Error (MAE) [Sebastiani, 2019]. MAE is the average of absolute differences between true ($\vec{q}$) and predicted ($\hat{\vec{q}}$) quantification for a set of classes $\mathcal{C}$. The performances were compared using the Friedman test with 95% confidence level and Nemenyi post hoc test.

Table 2 presents a brief description of the datasets used in our experiments obtained from UCI [Dheeru and Karra Taniskidou, 2017], OpenML [Vanschoren *et al.*, 2013], and Reis [dos Reis *et al.*, 2018b] repositories. Specific citations are requested for HTRU2 [Lyon *et al.*, 2016], Mozilla4 [Koru *et al.*, 2007], Nomao [Candillier and Lemaire, 2012], and Occupancy Detection [Candanedo and Feldheim, 2016].

A couple of observations about the datasets are due. First, Wine Type dataset is similar to Wine Quality. However, we want to differentiate between white and red wines, rather than

Dataset	Size	Features	Repository
AedesSex	24,000	27	Reis
Anuran Calls	6,585	22	UCI
ArabicDigit	8,800	27	UCI
BNG (vote)	39,366	9	OpenML
Spambase	4,601	57	UCI
Handwritten-QG	4,014	63	Reis
Wine Type	6,497	12	UCI
Wine Quality	6,497	12	UCI
HTRU2	17,898	8	UCI
Letter Recognition	20,000	16	UCI
Mozilla4	15,545	5	OpenML
Nomao	34,465	118	OpenML
Occupancy Detection	20,560	5	UCI

Table 2: Datasets description.

the wine quality. Second, ArabicDigit is a preprocessed version of the original so that all examples have the same number of features [dos Reis *et al.*, 2018b], and the objective is to predict the sex of the speaker rather than which digit is spoken.

7 Experimental Evaluation

We open this section by showing the quantification results for all algorithms and datasets, considering only our largest test set size of 500 observation points. Figure 3 shows the critical difference diagram.

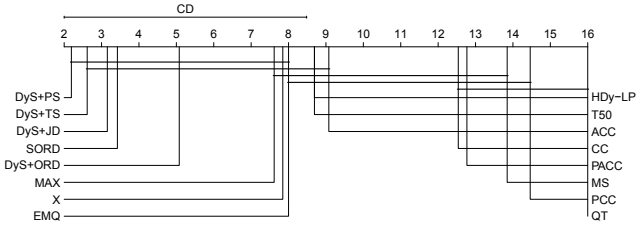

Figure 3: Nemenyi's post hoc test for mean absolute quantification error. Test sample composed of 500 cases. Groups of methods which are not significantly different at $p < 0.05$ are connected.

According to the results, algorithms based on distribution matching (III - Taxonomic Group) achieve the best performance. Particularly, four instances of the DyS framework are among the top five quantifiers. These results are not surprising, given the nature of these quantifiers, which are benefited by test sets with a large number of instances, allowing them to model the data distribution precisely.

These results are a typical summary of the experimental results we can find in quantification literature. In short, a set of algorithms compared over a collection of datasets for an arbitrary test set size. However, as discussed in Section 1, such results would only be representative of the whole spectrum of test set sizes if the performance of the quantification algorithms were independent of such a variable.

To illustrate that this assumption does not hold, Figure 4 shows the mean absolute error for all test set sizes and class distributions, for the best-ranked algorithm according to the previous results in Figure 3. Due to lack of space, our supplemental material website contains figures for all other algorithms[1].

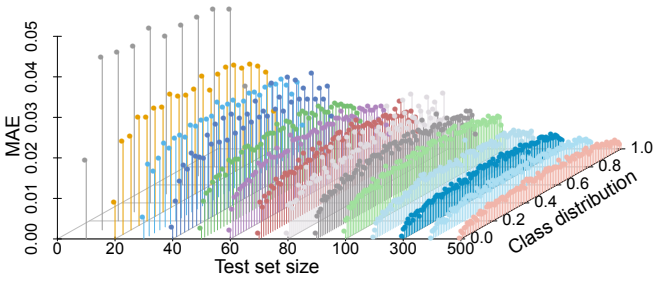

Figure 4: Mean absolute error variability for test set size and class distribution variability for DyS+PS.

Figure 4 exhibits small errors across all class distributions for test set sizes with more than 100 instances. However, for smaller test sets, with less than 100 cases, the errors tend to increase quickly. A worrisome aspect of these results is the fact that the literature on quantification has mostly ignored small test sets in their evaluations.

Motivated by our initial results, we reapply the same statistical analysis for all algorithms and a tiny test set size of ten observations. Figure 5 shows the critical difference diagram for all quantification approaches. The rank of the algorithms has significantly changed compared to the ranking built with test sets of 500 observations.

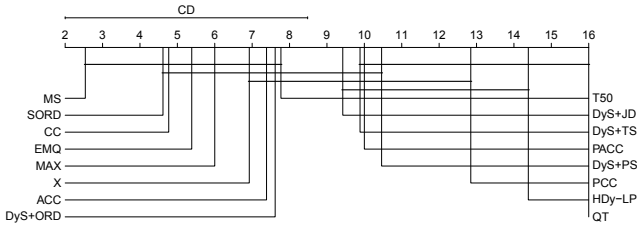

Figure 5: Nemenyi's post hoc test for mean absolute quantification error. Test sample composed of 10 cases. Groups of methods which are not significantly different at $p < 0.05$ are connected.

For large test set sizes, instances of the framework DyS produced the best results followed by MAX and X, which are algorithms from threshold selection family (I - Taxonomic Group). We also note that HDy-LP performed poorly, albeit it being similar to DyS. This behavior was reported by Maletzke et al. [2019] as well, where the authors noted the influence of the number of bins used to describe score distributions.

Conversely, MS was the best-ranked algorithm for the smallest test set size followed by SORD (which is also one instance of DyS), and CC. In the literature, MS is reported as one of the best quantification algorithms while the CC is considered the naive strategy and frequently regarded as a baseline. We also note that EMQ performs better in small test sets than in large ones. On the other hand, PCC, PACC, and QT perform poorly in all scenarios (large and small test set sizes). For QT algorithm, we used source code provided by the authors [Milli *et al.*, 2013].

Moreover, SORD and DyS+ORD were the only instances of DyS with no statistical difference against MS when the test set size is composed of ten cases. The explanation is that

ORD is less affected by sparse histograms than other distance functions, since it can compare misaligned histogram bins. SORD does not use histograms at all.

Another interesting finding is that there is statistical difference between DyS+PS and MS in both scenarios, even though DyS+PS was the best performing algorithm in one setting, and MS was the best in the other one. Nevertheless, we must keep in mind that these results were obtained for several datasets from different domains. Each dataset can be analyzed separately. For the sake of clarity, we include additional results in our supplemental material website[1].

Our results demonstrate that the experimental evaluations conducted until now by the literature have an incomplete analysis, since they neglect a relevant aspect of real-world problems. Consequently, current analyses provide inaccurate recommendations for quantification tasks.

In our subsequent analysis, we consider all quantification algorithms and all test set sizes. Figure 6 shows the ranking of all algorithms. Instances of the framework DyS perform better than other methods, except when it is using ORD as similarity function. As previously said, DyS+ORD, differently from the other instances, is less affected by sparseness, which seems less relevant for large test sets.

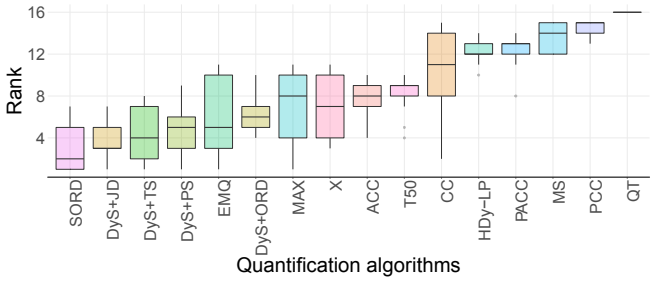

Figure 6: Aggregation of several rank positions for all quantification algorithms. Test sample size varied from 10 to 500.

The statistical analysis reveals that four of the five DyS instances evaluated are ranked as top quantifiers when we vary the test set size. SORD is the best quantifier, but it only performs better than HDy-LP, PACC, MS, PCC, and QT with a statistical difference ($p - value < 0.05$).

At last, we apply our proposed meta-learning framework, MLQ, to select one of the following quantifiers: (*i*) MS, which is the top-ranked for small test sets (Figure 5); (*ii*) DyS-PS, the best ranked on large test set size (Figure 3); and (*iii*) SORD, which has performed better than others when test set size has varied (Figure 6). Figure 7 shows the ranking of all algorithms when the test set size varies, including the MLQ.

Our meta-learning framework improves on the best quantifiers and outperform them in a more broad scenario, where test sample size varies. This achievement corroborates with our initial hypothesis. We have argued that the test set size should be considered in the experimental evaluation of quantifiers, guiding practitioners to make the best choice regarding quantification algorithm for each domain.

Although MLQ has decreased quantification errors when test set size varies, it is unclear how helpful was the meta-learning scheme. Ultimately, we fed the method with overtly simple metafeatures and the best quantifiers for each scenario. To obtain a more conclusive analysis, we investigate the real contribution of our meta-learning scheme and how far the MLQ results are from the ideal scenario. We define the *topline* (TOP) and *baseline* (RND) schemes. The *topline* scheme selects the best quantifier for each test set and the *baseline* makes this selection randomly. Figure 8 shows the mean absolute error for MLQ, *topline*, and *baseline* schemes.

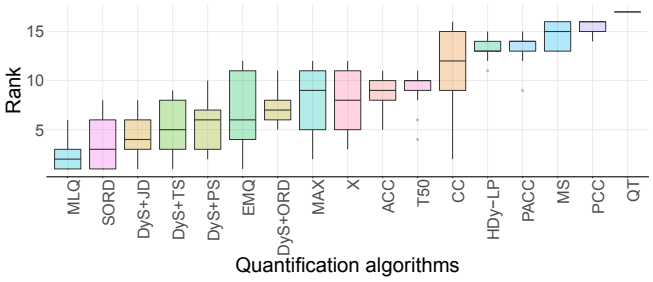

Figure 7: Aggregation of several rank positions for all quantifiers, including MLQ framework. Test sample size varied from 10 to 500.

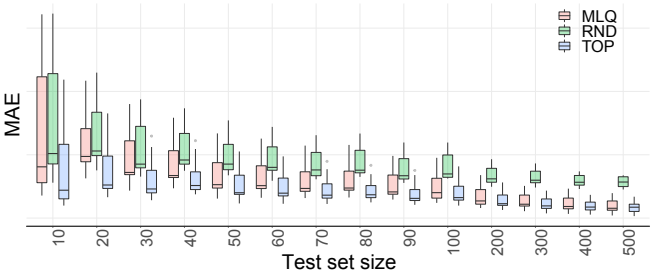

Figure 8: Mean absolute quantification error of MLQ, TOP, and RND. For test set varied from 10 to 500.

As expected the *topline* (TOP) performs better than our MLQ and the *baseline*. Our proposal outperforms the *baseline*, indicating that our meta-learning scheme learned how to properly select a quantification model, even though it can potentially be improved to get closer to the *topline*.

8 Conclusion

This paper is the first to question the relevance of the test set size in the design and evaluation of quantification methods, showing that a newly-proposed algorithm will be incomplete without analyzing its performance with different test set sizes. We also propose a simple strategy that uses meta-learning to select the best quantifier. Our proposal uses the test set size to perform a recommendation and can outperform the best single quantification method. Our future efforts will deal with the impact of the score quality on quantification methods.

Acknowledgments

The authors thank CAPES (PROEX-6909543/D), CNPq-TWAS fellow (139467/2017-3), FAPESP (2017/22896-7) and USAID (AID-OAA-F-16-00072).

References

[Barranquero *et al.*, 2013] Jose Barranquero, Pablo González, Jorge Díez, and Juan José del Coz. On the study of nearest neighbor algorithms for prevalence estimation in binary problems. *Pattern Recognition*, 46(2):472–482, February 2013.

[Beijbom *et al.*, 2015] Oscar Beijbom, Judy Hoffman, Evan Yao, Trevor Darrell, Alberto Rodriguez-Ramirez, Manuel Gonzalez-Rivero, and Ove Hoegh Guldberg. Quantification in-the-wild: data-sets and baselines. *arXiv preprint arXiv:1510.04811*, 2015.

[Bella *et al.*, 2010] Antonio Bella, Cesar Ferri, José Hernández-Orallo, and Maria Jose Ramirez-Quintana. Quantification via probability estimators. In *IEEE International Conference on Data Mining*, pages 737–742. IEEE, 2010.

[Brazdil *et al.*, 2009] Pavel Brazdil, Christophe Giraud-Carrier, Carlos Soares, and Ricardo Vilalta. Metalearning for algorithm recommendation: an introduction. *Metalearning: Applications to Data Mining*, pages 11–29, 2009.

[Candanedo and Feldheim, 2016] Luis M Candanedo and Véronique Feldheim. Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. *Energy and Buildings*, 112:28–39, 2016.

[Candillier and Lemaire, 2012] Laurent Candillier and Vincent Lemaire. Design and analysis of the nomao challenge active learning in the real-world. In *Active Learning in Real-world Applications, Workshop ECML-PKDD*, 2012.

[Chan and Ng, 2006] Yee Seng Chan and Hwee Tou Ng. Estimating class priors in domain adaptation for word sense disambiguation. In *International Conference on Computational Linguistics*, ACL-44, pages 89–96. Association for Computational Linguistics, 2006.

[Chen *et al.*, 2014] Yanping Chen, Adena Why, Gustavo Batista, Agenor Mafra-Neto, and Eamonn Keogh. Flying insect classification with inexpensive sensors. *Journal of insect behavior*, 27(5):657–677, 2014.

[Dheeru and Karra Taniskidou, 2017] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[dos Reis *et al.*, 2018a] Denis dos Reis, André Maletzke, Everton Cherman, and Gustavo Batista. One-class quantification. In *European Conference on Machine Learning*, pages 564–575, Dublin, 2018.

[dos Reis *et al.*, 2018b] Denis dos Reis, André Maletzke, Diego F. Silva, and Gustavo Batista. Classifying and counting with recurrent contexts. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'18, pages 1983–1992. ACM, 2018.

[Forman, 2005] George Forman. Counting positives accurately despite inaccurate classification. In *European Conference on Machine Learning*, pages 564–575. Springer, 2005.

[Forman, 2006] George Forman. Quantifying trends accurately despite classifier error and class imbalance. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 157–166. ACM, 2006.

[González *et al.*, 2017a] Pablo González, Alberto Castaño, Nitesh V Chawla, and Juan José Del Coz. A review on quantification learning. *ACM Computing Surveys (CSUR)*, 50(5):74, 2017.

[González *et al.*, 2017b] Pablo González, Jorge Díez, Nitesh Chawla, and Juan José del Coz. Why is quantification an interesting learning problem? *Progress in Artificial Intelligence*, 6(1):53–58, 2017.

[González-Castro *et al.*, 2013] Víctor González-Castro, Rocío Alaiz-Rodríguez, and Enrique Alegre. Class distribution estimation based on the hellinger distance. *Information Sciences*, 218:146 – 164, 2013.

[Koru *et al.*, 2007] A Gunes Koru, Dongsong Zhang, and Hongfang Liu. Modeling the effect of size on defect proneness for open-source software. In *Predictor Models in Software Engineering, 2007. PROMISE'07: ICSE Workshops 2007. International Workshop on*, pages 10–10. IEEE, 2007.

[Lemke *et al.*, 2015] Christiane Lemke, Marcin Budka, and Bogdan Gabrys. Metalearning: a survey of trends and technologies. *Artificial intelligence review*, 44(1):117–130, 2015.

[Lyon *et al.*, 2016] Robert J Lyon, BW Stappers, S Cooper, JM Brooke, and JD Knowles. Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 459(1):1104–1123, 2016.

[Maletzke *et al.*, 2018a] Andre Maletzke, Claudia Milare, Barbara Nadai, Jesse Saroli, Shailendra Singh, Juliano Corbi, Agenor Mafra-Neto, Eamonn Keogh, and Gustavo Batista. Automatic insect recognition with optical sensors with variability of temperature and humidity. In *AMCA 84th Annual Meeting, 2018.*, 2018.

[Maletzke *et al.*, 2018b] André G Maletzke, Denis dos Reis, and Gustavo Batista. Combining instance selection and self-training to improve data stream quantification. *Journal of the Brazilian Computer Society*, 24(1):12, 2018.

[Maletzke *et al.*, 2019] André Maletzke, Denis dos Reis, Everton Cherman, and Gustavo Batista. Dys: a framework for mixture models in quantification. In *AAAI Conference on Artificial Intelligence*, AAAI '19, 2019.

[Milli *et al.*, 2013] Letizia Milli, Anna Monreale, Giulio Rossetti, Fosca Giannotti, Dino Pedreschi, and Fabrizio Sebastiani. Quantification trees. In *International Conference on Data Mining*, pages 528–536. IEEE, 2013.

[Pérez-Gállego *et al.*, 2019] Pablo Pérez-Gállego, Alberto Castaño, José Ramón Quevedo, and Juan José del Coz. Dynamic ensemble selection for quantification tasks. *Information Fusion*, 45:1–15, 2019.

[Saerens *et al.*, 2002] Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural computation*, 14(1):21–41, 2002.

[Sebastiani, 2019] Fabrizio Sebastiani. Evaluation measures for quantification: An axiomatic approach. *Information Retrieval Journal*, pages 1–34, 2019.

[Silva *et al.*, 2015] Diego F Silva, Vinícius MA Souza, Daniel PW Ellis, Eamonn J Keogh, and Gustavo Batista. Exploring low cost laser sensors to identify flying insect species. *Journal of Intelligent & Robotic Systems*, 80(1):313–330, 2015.

[Vanschoren *et al.*, 2013] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2013.

[Vapnik, 2013] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.