Re-Assessing the "Classify and Count" Quantification Method

Alejandro Moreo[0000−0002−0377−1025] and Fabrizio Sebastiani[0000−0003−4221−6427]

Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
firstname.lastname@isti.cnr.it

Abstract. *Learning to quantify* (a.k.a. *quantification*) is a task concerned with training unbiased estimators of class prevalence via supervised learning. This task originated with the observation that "Classify and Count" (CC), the trivial method of obtaining class prevalence estimates, is often a biased estimator, and thus delivers suboptimal quantification accuracy; following this observation, several methods for learning to quantify have been proposed that have been shown to outperform CC. In this work we contend that previous works have failed to use properly optimised versions of CC. We thus reassess the real merits of CC (and its variants), and argue that, while still inferior to some cutting-edge methods, they deliver near-state-of-the-art accuracy once (a) hyperparameter optimisation is performed, and (b) this optimisation is performed by using a true quantification loss instead of a standard classification-based loss. Experiments on three publicly available binary sentiment classification datasets support these conclusions.

Keywords: Learning to quantify · Quantification · Prevalence estimation · Classify and count

1 Introduction

Learning to quantify (a.k.a. *quantification*) consists of training a predictor that returns estimates of the relative frequency (a.k.a. *prevalence*, or *prior probability*) of the classes of interest in a set of unlabelled data items, where the predictor is trained on a set of labelled data items [12]. When applied to text, quantification is important for several applications, e.g., gauging the collective satisfaction for a certain product from textual comments [7], establishing the popularity of a given political candidate from blog posts [16], predicting the amount of consensus for a given governmental policy from tweets [4], or predicting the amount of readers who will find a product review helpful [5].

The rationale of this task is that many real-life applications of text classification suffer from *distribution shift* [21], the phenomenon according to which the distribution $p_U(y)$ of the labels in the set of unlabelled test documents U is different from the distribution $p_L(y)$ that the labels have in the set of labelled training documents L. It has been argued that, in the presence of distribution shift, the trivial strategy of using a standard classifier to classify all the unlabelled documents in U and counting the number of documents that have been assigned to the class (the "Classify and Count" (CC) method), delivers poor class prevalence estimates. The reason is that most supervised learning methods are based on the IID assumption, which implies (among others) that the distribution of the labels is the same in L and U. "Classify and Count" is considered a *biased estimator* of class prevalence, since the goal of standard classifiers is to minimise (assuming for simplicity a binary setting) *classification* error measures such as (FP+FN), while the goal of a quantifier should be to minimise *quantification* error measures such as $|\mathrm{FP} - \mathrm{FN}|$. (In this paper we tackle binary quantification, so FP and FN represent the numbers of false positives and false negatives, respectively, from a binary contingency table.) Following this observation, several methods for learning to quantify have been proposed, and have been experimentally shown to outperform CC.

In this paper we contend that previous works, when testing advanced quantification methods, have used, as baselines, versions of CC that had not been properly optimised. This means that our knowledge of the relative merits of CC and other supposedly more advanced methods is still unreliable. We thus reassess the real merits of CC by running extensive experiments (on three publicly available sentiment classification datasets) in which we compare properly optimised versions of CC and its three main variants (PCC, ACC, PACC) with a number of more advanced quantification methods. In this experimentation we properly optimise all quantification methods, i.e., (a) we optimise their hyperparameters, and (b) we conduct this optimisation by minimising a quantification loss rather than a classification loss. Our results indicate that, while still inferior to some cutting-edge quantification methods, CC and its variants deliver near-state-of-the-art quantification accuracy once hyperparameter optimisation is performed properly. We make available all the code and the datasets that we have used for our experiments[1]; this allows our results to be easily reproduced by other researchers.

The rest of this paper is structured as follows. In Section 2 we briefly present Classify and Count and its three main variants. In Section 3 we discuss how proper optimisation of quantification methods should be performed, and show that these standards have seldom been adhered to in past quantification literature when using CC as a baseline. Section 4 discusses the new experiments we have run in order to compare CC with other methods under more strict experimental standards. Section 4.4 discusses the results and the conclusions that they allow to draw.

2 "Classify and Count" and its variants

In this paper we will use the following notation. We assume a binary setting, with the two classes $\mathcal{Y} = \{\oplus, \ominus\}$, standing for Positive and Negative. By $\mathbf{x}$ we indicate a document drawn from a domain $\mathcal{X}$ of documents; by $L \subset \mathcal{X}$ we denote a set of labelled documents, that we typically use as a training set, while by U we denote a sample of unlabelled documents, that we typically use as the sample to quantify on. By $p_y(\sigma)$ we indicate the true prevalence of class y in sample σ, by $\hat{p}_y(\sigma)$ we indicate an estimate of this prevalence[2], and by $\hat{p}_y^M(\sigma)$ we indicate the estimate of this prevalence as obtained via quantification method M.

An obvious way to solve quantification is by aggregating the scores assigned by a classifier to the unlabelled documents. We first define two different aggregation methods, one that uses a "hard" classifier (i.e., a classifier $h_\oplus : \mathcal{X} \rightarrow \{0,1\}$ that outputs binary decisions, 0 for $\ominus$ and 1 for $\oplus$) and one that uses a "soft" classifier (i.e., a classifier $s_\oplus : \mathcal{X} \rightarrow [0,1]$ that outputs posterior probabilities $\Pr(\oplus|\mathbf{x})$, representing the probability that the classifier attributes to the fact that $\mathbf{x}$ belongs to the $\oplus$ class). Of course, $\Pr(\ominus|\mathbf{x}) = (1 - \Pr(\oplus|\mathbf{x}))$.

The (trivial) *classify and count* quantifier (CC) then comes down to computing

$$\hat{p}_\oplus^{\mathrm{CC}}(U) = \frac{\sum_{\mathbf{x} \in U} h_\oplus(\mathbf{x})}{|U|} \tag{1}$$

[1] `https://github.com/AlexMoreo/CC`

[2] Consistently with most mathematical literature, we use the caret symbol (ˆ) to indicate estimation.

while the *probabilistic classify and count* quantifier (PCC [3]) is defined by

$$\hat{p}^{\mathrm{PCC}}_{\oplus}(U) = \frac{\sum_{\mathbf{x}\in U} s_{\oplus}(\mathbf{x})}{|U|} \tag{2}$$

Of course, for any method M we have $\hat{p}^{M}_{\ominus}(U) = (1 - \hat{p}^{M}_{\oplus}(U))$.

A popular quantification method consists of applying an *adjustment* to the $\hat{p}^{\mathrm{CC}}_{\oplus}(U)$ prevalence estimates. It is easy to check that, in the binary case, the true prevalence $p_{\oplus}(U)$ and the estimated prevalence $\hat{p}_{\oplus}(U)$ are such that

$$p_{\oplus}(U) = \frac{\hat{p}^{\mathrm{CC}}_{\oplus}(U) - \mathrm{FPR}_h}{\mathrm{TPR}_h - \mathrm{FPR}_h} \tag{3}$$

where TPR_h and FPR_h stand for the *true positive rate* and *false positive rate* that the classifier $h_{\oplus}$ used to obtain $\hat{p}^{\mathrm{CC}}_{\oplus}$ has on U. The values of TPR_h and FPR_h are unknown, but can be estimated via k-fold cross-validation on the training data. In the binary case this comes down to using the results $h_{\oplus}(\mathbf{x})$ obtained in the k-fold cross-validation (i.e., when $\mathbf{x}$ ranges on the training documents) in equations

$$\hat{\mathrm{TPR}}_h = \frac{\sum_{\mathbf{x}\in\oplus} h_{\oplus}(\mathbf{x})}{|\{\mathbf{x}\in\oplus\}|} \qquad \hat{\mathrm{FPR}}_h = \frac{\sum_{\mathbf{x}\in\ominus} h_{\oplus}(\mathbf{x})}{|\{\mathbf{x}\in\ominus\}|} \tag{4}$$

We obtain $\hat{p}^{\mathrm{ACC}}_{\oplus}(U)$ estimates, which define the *adjusted classify and count* method [10] (ACC), by replacing TPR_h and FPR_h in Equation 3 with the estimates of Equation 4, i.e.,

$$\hat{p}^{\mathrm{ACC}}_{\oplus}(U) = \frac{\hat{p}^{\mathrm{CC}}_{\oplus}(U) - \hat{\mathrm{FPR}}_h}{\hat{\mathrm{TPR}}_h - \hat{\mathrm{FPR}}_h} \tag{5}$$

If the soft classifier $s_{\oplus}(\mathbf{x})$ is used in place of $h_{\oplus}(\mathbf{x})$, analogues of $\hat{\mathrm{TPR}}_h$ and $\hat{\mathrm{FPR}}_h$ from Equation 4 can be defined as

$$\hat{\mathrm{TPR}}_s = \frac{\sum_{\mathbf{x}\in\oplus} s_{\oplus}(\mathbf{x})}{|\{\mathbf{x}\in\oplus\}|} \qquad \hat{\mathrm{FPR}}_s = \frac{\sum_{\mathbf{x}\in\ominus} s_{\oplus}(\mathbf{x})}{|\{\mathbf{x}\in\ominus\}|} \tag{6}$$

We obtain $\hat{p}^{\mathrm{PACC}}_{\oplus}(U)$ estimates, which define the *probabilistic adjusted classify and count* method (PACC [3]), by replacing all factors in the right-hand side of Equation 5 with their "soft" counterparts from Equations 2 and 6, i.e.,

$$\hat{p}^{\mathrm{PACC}}_{\oplus}(U) = \frac{\hat{p}^{\mathrm{PCC}}_{\oplus}(U) - \hat{\mathrm{FPR}}_s}{\hat{\mathrm{TPR}}_s - \hat{\mathrm{FPR}}_s} \tag{7}$$

3 Quantification and parameter optimisation

3.1 Unsuitable parameter optimisation and weak baselines

The reason why we here reassess CC and its variants is that we believe that, in previous papers where these methods have been used as baselines, their full potential has not been realised because of missing or unsuitable optimisation of the hyperparameters of the classifier on which the method is based.

Specifically, both CC and its variants rely on the output of a previously trained classifier, and this output usually depends on some hyperparameters. Not only the quality of this output heavily depends on whether these hyperparameters have been optimised or not (on some held-out set or via k-fold cross-validation), but *it also depends on what evaluation measure this optimisation has used as a criterion for model selection.* In other words, given that hyperparameter optimisation chooses the value of the parameter that minimises error, it would make sense that, for a classifier to be used for quantification purposes, "error" is measured via a function that evaluates *quantification* error, and not classification error. Unfortunately, in most previous quantification papers, researchers either do not specify whether hyperparameter optimisation was performed at all [8,10,13,14,16,18,25,26], or leave the hyperparameters at their default values [1,3,9,15,20], or do not specify which evaluation measure they use in hyperparameter optimisation [7,11], or use, for this optimisation, a classification-based loss [2,24]. In retrospect, we too plead guilty, since some of the papers quoted here are our own.

All this means that CC and their variants, when used as baselines, have been turned into *weak* baselines, and this means that the merits of more modern methods relative to them have possibly been exaggerated, and are thus yet to be assessed reliably. In this paper we thus engage in a reproducibility study, and present results from text quantification experiments in which, contrary to the situations described in the paragraph above, we compare *carefully optimised* versions of CC and its variants with a number of (*carefully optimised* versions of) more modern quantification methods, in an attempt to assess the relative value of each in a robust way.

3.2 Quantification-oriented parameter optimisation

In order to perform quantification-oriented parameter optimisation we need to be aware that there may exist two types of parameters that require estimation and/or optimisation, i.e., (a) the hyperparameters of the classifier on which the quantification method is based, and (b) the parameters of the quantification method itself. Within all the quantification methods we consider in this paper, only ACC and PACC have their own parameters that need estimation, i.e., $\hat{\mathrm{TPR}}_h$ and $\hat{\mathrm{FPR}}_h$ (for ACC) and $\hat{\mathrm{TPR}}_s$ and $\hat{\mathrm{FPR}}_s$ (for PACC).

The way we perform parameter optimisation is the following. We assume that the dataset comes with a predefined split between a training set L and a test set U. (This assumption is indeed verified for the datasets we will be using in Section 4.) We first split L into a part L_{Tr} that will be used for training purposes and a part L_{Va} that will be used as a held-out validation set for optimising the parameters of the classifier. We then extract, from the validation set L_{Va}, several random validation samples, each characterised by a predefined prevalence of the $\oplus$ class; here, our goal is allowing the validation to be conducted on a variety of scenarios characterised by widely different values of class prevalence, and, as a consequence, by widely different amounts of distribution shift.[3] In order to do this, we extract each validation sample σ by randomly undersampling one or both classes in L_{Va}, in order to obtain a sample with prespecified class prevalence values. We draw samples with a desired prevalence value and a fixed amount q of documents; in order to achieve this, in some cases only one class needs to be undersampled while in some other cases this needs to happen for both classes. We use random sampling without replacement if the number of available examples of $\oplus$ (resp. $\ominus$) is greater or equal to the number of required ones, and with replacement otherwise. We extract samples with a prevalence of the $\oplus$ class in the set $\{\pi_1, ..., \pi_n\}$; for each of these n values we generate m random samples consisting of q validation documents each. Let Θ be the set of parameters that we are going to optimise. Given the established grid of value combinations $\theta_1, ..., \theta_n$ that we are going to test for Θ, for each θ_i we do the following, depending

[3] Note that this is similar to what we do, say, in classification, where the different parameter values are tested on many validation documents; here we test these parameter values on many validation *samples*, since the objects of study of text quantification are document samples inasmuch as the objects of study of text classification are individual documents.

on whether the quantification method has its own parameters (Case 1 below) or not (Case 2 below):

1. If the quantification method M we are going to optimise requires some parameters λ_i to be estimated, we first split L_{Tr} into a part $L^{\mathrm{Tr}}_{\mathrm{Tr}}$ and a part $L^{\mathrm{Va}}_{\mathrm{Tr}}$, training the classifier on $L^{\mathrm{Tr}}_{\mathrm{Tr}}$ using the chosen learner parameterised with θ_i, and estimate parameters λ_i on $L^{\mathrm{Va}}_{\mathrm{Tr}}$.[4] Among the variants of CC, this applies to methods ACC and PACC, which require the estimation of (the hard or soft version of) TPR and FPR. Other methods used in the experiments of Section 4 and that also require some parameter to be estimated are HDy and QuaNet (see Section 4.3.3).
2. If the quantification method M we are going to optimise does not have any parameter that requires estimation, then we train our classifier on L_{Tr}, using the chosen learner parameterised with θ_i, and use quantification method M on all the samples extracted from L_{Va}.

In both cases, we measure the quantification error via an evaluation measure for quantification that combines (e.g., averages) the results across all the validation samples. As our final value combination for parameter set Θ we choose the value combination θ_i for which quantification error is minimum.

Note that, in the above discussion, each time we split a labelled set into a training set and a validation set for parameter estimation / optimisation purposes, we could instead perform a k-fold cross-validation; the parameter estimation/optimisation would be more robust, but the computational cost of the entire process would be k times higher. While the latter method is also, from a methodological standpoint, an option, in this paper we stick to the former method, since the entire parameter optimisation process is, from a computational point of view, already very expensive.

4 Experiments

4.1 Datasets

In order to conduct our experiments we use the same datasets and experimental protocol as used in [6]. Specifically, we run our experiments on three sentiment classification datasets, i.e., (i) IMDB, the popular *Large Movie Review Dataset* [19]; (ii) KINDLE, a set of reviews of Kindle e-book readers [6], and (iii) HP, a set of reviews of the books from the Harry Potter series [6].[5] For all datasets we adopt the same split between training set L and test set U as in [6]. The IMDB, KINDLE, and HP datasets are examples of balanced, imbalanced, and severely imbalanced datasets, since the prevalence values of the $\oplus$ class in the training set L are 0.500, 0.917, 0.982, respectively. Some basic statistics from these datasets are reported in Table 1. We refer the reader to [6] for more details on the genesis of these datasets.

In our experiments, from each set of training data we randomly select 60% of the documents for training purposes, leaving the remaining 40% for the hyperparameter optimisation phase; these random splits are stratified, meaning that both splits present the same prevalence as the set that originated them. In this phase (see Section 3.2) we use $n = 21$, $m = 10$, and $q = 500$, i.e., we generate $m = 10$ random

[4] Note that we do *not* retrain the classifier on the entire L_{Tr}. While this might seem beneficial, since L_{Tr} contains more training data than $L^{\mathrm{Tr}}_{\mathrm{Tr}}$, we need to consider that the estimates $\hat{\mathrm{TPR}}_h$ and $\hat{\mathrm{FPR}}_h$ have been computed on L_{Tr} and not on $L^{\mathrm{Tr}}_{\mathrm{Tr}}$.

[5] The three datasets are available at `https://doi.org/10.5281/zenodo.4117827` in preprocessed form. The raw versions of the HP and KINDLE datasets can be accessed from `http://hlt.isti.cnr.it/quantification/`, while the raw version of IMDB can be found at `https://ai.stanford.edu/~amaas/data/sentiment/`.

Table 1. The three datasets used in our experiments; the columns indicate the class prevalence values of the $\oplus$ and $\ominus$ classes, and the numbers of documents contained in the training set L and the test set U.

	$\oplus$	$\ominus$	L	L_{Tr}	L_{Va}	U
IMDB	0.500	0.500	25,000	15,000	10,000	25,000
Kindle	0.917	0.083	3,821	2,292	1,529	21,592
HP	0.982	0.018	9,533	5,720	3,813	18,401

samples of $q = 500$ document each, for each of the $n = 21$ prevalence values of the $\oplus$ class in $\{0.00, 0.05, ..., 0.95, 1.00\}$.

In order to evaluate a quantifier over a wide spectrum of test prevalence values, we use essentially the same process that we have discussed in Section 3.2 for parameter optimisation; that is, along with [6,10], we randomly undersample one or both classes in the test set U in order to obtain a sample with specified class prevalence values in the test set. Here we generate $m = 100$ random samples of $q = 500$ document each, for each of the $n = 21$ prevalence values of the $\oplus$ class in $\{0.00, 0.05, ..., 0.95, 1.00\}$.

4.2 Evaluation measures

As the measures of quantification error we use *Absolute Error* (AE) and *Relative Absolute Error* (RAE), defined as

$$\text{AE}(p, \hat{p}) = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} |\hat{p}(y) - p(y)| \tag{8}$$

$$\text{RAE}(p, \hat{p}) = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} \frac{|\hat{p}(y) - p(y)|}{p(y)} \tag{9}$$

where $\mathcal{Y}$ is the set of classes of interest ($\mathcal{Y} = \{\oplus, \ominus\}$ in our case). Note that RAE is undefined when at least one of the classes $y \in \mathcal{Y}$ is such that its prevalence in U is 0. To solve this problem, in computing RAE we smooth both all $p(y)$'s and $\hat{p}(y)$'s via additive smoothing, i.e., we take $\underline{p}(y) = \frac{\epsilon + p(y)}{\epsilon|\mathcal{Y}| + \sum_{y \in \mathcal{Y}} p(y)}$, where $\underline{p}(y)$ denotes the smoothed version of $p(y)$ and the denominator is just a normalising factor (same for the $\underline{\hat{p}}(y)$'s); following [10], we use the quantity $\epsilon = \frac{1}{2|U|}$ as the smoothing factor. We then use the smoothed versions of $p(y)$ and $\hat{p}(y)$ in place of their original non-smoothed versions in Equation 9; as a result, RAE is always defined.

The reason why we use AE and RAE is that from a theoretical standpoint they are, as it has been recently argued [27], the most satisfactory evaluation measures for quantification.

4.3 The quantifiers

We here describe all the quantification systems we have used in this work.

4.3.1 Data processing. We preprocess our documents by using the stop word remover and default tokeniser available within the `scikit-learn` framework[6], and masking as unknown all terms occurring less than 5 times in the training set. In all three datasets we remove all punctuation marks and lowercase the text.

As the weighting criterion we use a version of the well-known tfidf method, i.e.,

$$\text{tfidf}(f, \mathbf{x}) = \log \#(f, \mathbf{x}) \times \log \frac{|L|}{|\mathbf{x}' \in L : \#(f, \mathbf{x}') > 0|} \tag{10}$$

[6] `http://scikit-learn.org/`

where $\#(f, \mathbf{x})$ is the raw number of occurrences of feature f in document $\mathbf{x}$; weights are then normalised via cosine normalisation, as

$$w(f, \mathbf{x}) = \frac{\text{tfidf}(f, \mathbf{x})}{\sqrt{\sum_{f'} \text{tfidf}(f', \mathbf{x})^2}} \tag{11}$$

Among the learners we use for classification (see below), the only one that does not rely on a tfidf-based representation is CNN. This learner simply converts all documents into lists of unique numeric IDs, indexing the terms in the vocabulary. We pad the documents to the first 300 words.

4.3.2 CC and its variants. In our experiments we generate versions of CC, ACC, PCC, and PACC, using five different learners, i.e., support vector machines (SVMs), logistic regression (LR), random forests (RFs), multinomial naive Bayes (MNB), and convolutional neural networks (CNN). For the first four learners we rely on the implementations available from `scikit-learn`, while the CNN deep neural network is something we have implemented ourselves using the `pytorch` framework[7]. The setups that we use for these learners are the following:

- SVM: We use soft-margin SVMs with linear kernel and L2 regularisation, and we explicitly optimise the C parameter (in the range $C \in \{10^i\}$ with $i \in [-4, -3, \ldots, 4, 5]$) that determines the tradeoff between the margin and the training error. We also optimise the $J_\oplus$ and $J_\ominus$ "rebalancing" parameters, which determine whether to impose that misclassifying a $\oplus$ document has a different cost than misclassifying a $\ominus$ document (in this case one sets $J_\oplus = \frac{p_\ominus(L)}{p_\oplus(L)}$ and $J_\ominus = 1$), or not (in this case one sets $J_\oplus = J_\ominus = 1$) [22].
- LR: We use L2 regularisation, we explicitly optimise the rebalancing parameters (as in SVMs) and the regularisation coefficient C.
- RF: we optimise the number of estimators in the range $\{10, 50, 100, 250, 500\}$, the max depth in $\{5, 15, 30\}$, and the splitting function in $\{\text{Gini}, \text{Entropy}\}$.
- MNB: We use Laplace smoothing, and we optimise the additive factor α in the range $\{0, 0.05, 0.1, \ldots, 0.95, 1\}$.
- CNN: we use a single convolutional layer with γ output channels for three window lengths of 3, 5, and 7 words. Each convolution is followed by a ReLU activation function and a max-pooling operation. All convolved outputs are then concatenated and processed by an affine transformation and a sigmoid activation that converts the outputs into posterior probabilities. We use the Adam optimiser (with learning rate $1E^{-3}$ and all other parameters at their default values) to minimise the balanced binary cross-entropy loss, set the batch size to 100, and train the net for 500 epochs, but we apply an early stop after 20 consecutive training epochs showing no improvement in terms of F_1 for the minority class on the validation set. We explore the dimensionality of the embedding space in the range $\{100, 300\}$, the number of output channels γ in $\{256, 512\}$, whether to apply dropout to the last layer (with a drop probability of 0.5) or not, and whether to apply weight decay (with a factor of $1E^{-4}$) or not.

Since we perform hyperparameter optimisation via grid search, the number of validations (i.e., combinations of hyperparameters) that we perform amounts to 10 for SVMs, 10 for LR, 30 for RF, 21 for MNB, and 16 for CNN.

In the following, by the notation M_l^m we will indicate quantification method M using learner l whose parameters have been optimised using measure m (where $M_l^{\emptyset}$ indicates that no optimisation at all has been carried out). We will test, on all three

[7] `https://pytorch.org/`

datasets, all combinations in which M ranges on {CC, ACC, PCC, PACC}, l ranges on {SVM, LR, RF, MNB, CNN}, and m ranges on $\{\mathrm{A}, \mathrm{F}_1, \mathrm{AE}\}$, where A denotes vanilla accuracy, F_1 is the well-known harmonic mean of precision and recall, and AE is absolute error. We stick to the tradition of computing F_1 with respect to the minority class, which always turns out to be $\ominus$ in all three datasets (this means that, e.g., the true positives of the contingency table are the documents that the classifier assigns to $\ominus$ and that indeed belong to $\ominus$).

Note that PCC requires the classifier to return posterior probabilities. Since SVMs does not produce posterior probabilities, for $\mathrm{PCC}_{\mathrm{SVM}}$ and $\mathrm{PACC}_{\mathrm{SVM}}$ we calibrate the confidence scores that SVMs output by using Platt's method [23].

4.3.3 Advanced quantification methods. As the advanced methods that we test against CC and its variants, we use a number of more sophisticated systems that have been top-performers in the recent quantification literature.

- We use the method of [26] (that we here call EMQ, standing for *Expectation Maximisation for Quantification*), which consists of training a probabilistic classifier and then exploiting the EM algorithm to iteratively shift the estimation of $p_y(\sigma)$ from the one obtained from the training set to one that maximises the likelihood on the test data. As the underlying learner for EMQ we use LR, since (as MNB) it returns posterior probabilities (which EMQ needs), since these probabilities tend to be (differently from those returned by MNB) well-calibrated, and since it is well-known to perform much better than MNB.
- We use methods SVM(KLD), SVM(NKLD), SVM(Q), SVM(AE), SVM(RAE), from the "structured output learning" camp. Each of them is the result of instantiating the $\mathrm{SVM}_{\mathrm{perf}}$ structured output learner [17] to optimise a different loss function. SVM(KLD) [9] minimises the Kullback-Leibler Divergence (KLD); SVM(NKLD) [8] minimises a version of KLD normalised via the logistic function; SVM(Q) [1] minimises the harmonic mean of a classification-oriented loss (recall) and a quantification-oriented loss (RAE). We also add versions that minimise AE and RAE, since these latter are now, as indicated in Section 4.2, the evaluation measures for quantification considered most satisfactory, and the two used in this paper for evaluating the quantification accuracy of our systems. We optimise the C parameter of $\mathrm{SVM}_{\mathrm{perf}}$ in the range $C \in \{10^i\}$, with $i \in [-4, -3, \ldots 4, 5]$. In this case we do not optimise the $J_\oplus$ and $J_\ominus$ "rebalancing" parameters since this option is not available in $\mathrm{SVM}_{\mathrm{perf}}$.
- We use the HDy method of [14]. The method searches for the prevalence values that minimise the divergence (as measured via the Hellinger Distance) between two cumulative distributions of posterior probabilities returned by the classifier, one for the unlabelled examples and the other for a validation set. The latter is a mixture of the distributions of posterior probabilities returned for the $\oplus$ and $\ominus$ validation examples, respectively, where the parameters of the mixture are the sought class prevalences. HDy needs no further hyperparameter exploration beyond those of the classifier. We use LR as the classifier for the same reasons discussed for EMQ.
- We use the QuaNet system, a "meta-"quantification method based on deep learning [6]. QuaNet takes as input a list of document embeddings, together with and sorted by the classification scores returned by a classifier. A bidirectional LSTM processes this list and produces a quantification embedding that is then concatenated with a vector of predictions produced by an ensemble of simpler quantification methods (we here employ CC, ACC, PCC, PACC, and EMQ). The resulting vector passes through a set of fully connected layers (followed by ReLU activations and dropout) that return the estimated class prevalence values. We use CNN as the learner since, among the learners we use in this paper,

Table 2. Results showing how the quantification error of CC changes according to the measure used in hyperparameter optimisation; a negative percentage indicates a reduction in error with respect to using the method with default parameters. The background cell colour indicates improvement (green) or deterioration (red), while its intensity is proportional to the absolute magnitude of this improvement/deterioration.

	IMDB		KINDLE		HP	
	AE	RAE	AE	RAE	AE	RAE
$\mathrm{CC}^{\varnothing}_{\mathrm{SVM}}$	0.065	6.029	0.305	15.928	0.471	24.058
$\mathrm{CC}^{\mathrm{A}}_{\mathrm{SVM}}$	0.059 (-9.6%)	5.408 (-10.3%)	0.245 (-19.8%)	13.220 (-17.0%)	0.401 (-14.9%)	20.645 (-14.2%)
$\mathrm{CC}^{\mathrm{F}_1}_{\mathrm{SVM}}$	0.059 (-9.5%)	5.523 (-8.4%)	0.108 (-64.5%)	7.192 (-54.8%)	0.236 (-50.0%)	13.590 (-43.5%)
$\mathrm{CC}^{\mathrm{AE}}_{\mathrm{SVM}}$	0.065 (+0.3%)	6.091 (+1.0%)	0.100 (-67.1%)	7.555 (-52.6%)	0.119 (-74.8%)	10.593 (-56.0%)
$\mathrm{CC}^{\varnothing}_{\mathrm{LR}}$	0.059	5.477	0.470	23.990	0.500	25.508
$\mathrm{CC}^{\mathrm{A}}_{\mathrm{LR}}$	0.062 (+6.0%)	5.839 (+6.6%)	0.202 (-57.0%)	11.215 (-53.3%)	0.451 (-9.8%)	23.035 (-9.7%)
$\mathrm{CC}^{\mathrm{F}_1}_{\mathrm{LR}}$	0.062 (+5.3%)	5.725 (+4.5%)	0.163 (-65.3%)	9.278 (-61.3%)	0.229 (-54.3%)	13.505 (-47.1%)
$\mathrm{CC}^{\mathrm{AE}}_{\mathrm{LR}}$	0.062 (+6.1%)	5.745 (+4.9%)	0.094 (-80.0%)	7.087 (-70.5%)	0.110 (-78.0%)	10.304 (-59.6%)
$\mathrm{CC}^{\varnothing}_{\mathrm{RF}}$	0.155	13.388	0.448	22.988	0.493	25.196
$\mathrm{CC}^{\mathrm{A}}_{\mathrm{RF}}$	0.082 (-47.0%)	7.643 (-42.9%)	0.476 (+6.4%)	24.370 (+6.0%)	0.500 (+1.4%)	25.508 (+1.2%)
$\mathrm{CC}^{\mathrm{F}_1}_{\mathrm{RF}}$	0.083 (-46.7%)	7.739 (-42.2%)	0.474 (+5.9%)	24.267 (+5.6%)	0.499 (+1.2%)	25.458 (+1.0%)
$\mathrm{CC}^{\mathrm{AE}}_{\mathrm{RF}}$	0.081 (-47.5%)	7.589 (-43.3%)	0.481 (+7.5%)	24.590 (+7.0%)	0.500 (+1.4%)	25.508 (+1.2%)
$\mathrm{CC}^{\varnothing}_{\mathrm{MNB}}$	0.096	8.147	0.500	25.513	0.500	25.510
$\mathrm{CC}^{\mathrm{A}}_{\mathrm{MNB}}$	0.098 (+1.6%)	8.529 (+4.7%)	0.443 (-11.4%)	22.641 (-11.3%)	0.499 (-0.2%)	25.459 (-0.2%)
$\mathrm{CC}^{\mathrm{F}_1}_{\mathrm{MNB}}$	0.097 (+0.8%)	8.311 (+2.0%)	0.444 (-11.3%)	22.731 (-10.9%)	0.499 (-0.2%)	25.470 (-0.2%)
$\mathrm{CC}^{\mathrm{AE}}_{\mathrm{MNB}}$	0.097 (+0.9%)	8.431 (+3.5%)	0.443 (-11.4%)	22.701 (-11.0%)	0.499 (-0.2%)	25.464 (-0.2%)
$\mathrm{CC}^{\varnothing}_{\mathrm{CNN}}$	0.072	6.683	0.087	8.138	0.255	17.042
$\mathrm{CC}^{\mathrm{A}}_{\mathrm{CNN}}$	0.073 (+2.0%)	6.620 (-1.0%)	0.107 (+23.8%)	8.680 (+6.7%)	0.159 (-37.5%)	14.255 (-16.4%)
$\mathrm{CC}^{\mathrm{F}_1}_{\mathrm{CNN}}$	0.078 (+8.7%)	7.142 (+6.9%)	0.085 (-2.2%)	7.951 (-2.3%)	0.149 (-41.5%)	14.030 (-17.7%)
$\mathrm{CC}^{\mathrm{AE}}_{\mathrm{CNN}}$	0.074 (+3.2%)	6.613 (-1.0%)	0.109 (+26.2%)	8.591 (+5.6%)	0.343 (+34.3%)	19.008 (+11.5%)

it is the only one that returns both posterior probabilites and document embeddings (we use the last layer of the CNN as the document embedding). We set the hidden size of the bidirectional LSTM to $128 + 128 = 256$ and use two stacked layers. We also set the hidden sizes of the fully connected layers to 1024 and 512, and the dropout probability to 0.5. We train the network for 500 epochs, but we apply early stopping with a patience of 10 consecutive validations without improvements in terms of AE. Each training epoch consists of 200 quantification predictions, each of which for a batch of 500 randomly drawn documents at a prevalence sampled from the uniform distribution. In our case, validation epochs correspond to 21 quantification predictions for batches of 500 documents randomly sampled to have prevalence values $0.00, 0.05, \ldots, 0.95, 1.00$. We use Adam as the optimiser, with default parameters, to minimise mean square error. In order to train QuaNet, we split the training set L_{Tr} in three sets $L^{\mathrm{C-Tr}}_{\mathrm{Tr}}$, for training the classifier; $L^{\mathrm{Q-Tr}}_{\mathrm{Tr}}$, for training QuaNet; and $L^{\mathrm{Q-Va}}_{\mathrm{Tr}}$, for validating QuaNet. For this we use a 40%/40%/20% stratified split. When optimising QuaNet we do not explore any additional hyperparameter apart from those for the CNN.

– We also report results for *Maximum Likelihood Probability Estimation* (MLPE), the trivial baseline for quantification which makes the IID assumption and thus simply assumes that $p_{\oplus}(U)$ is identical to the training prevalence $p_{\oplus}(L)$ irrespectively of the set U.

Note that ACC, PACC, HDy, and QuaNet need to estimate their own parameters on a validation set, which means that their performance depends on exactly which documents this set consists of. In order to mitigate the impact of this random choice, for these methods we run each experiment 10 times. The results we report are the average scores across these 10 runs.

Table 3. Same as Table 2, but with ACC instead of CC.

	IMDB		Kindle		HP	
	AE	RAE	AE	RAE	AE	RAE
$\text{ACC}^{\emptyset}_{\text{SVM}}$	0.023	1.084	0.068	2.958	0.341	17.350
$\text{ACC}^{\text{A}}_{\text{SVM}}$	0.019 (-17.6%)	0.889 (-18.0%)	0.070 (+4.1%)	3.093 (+4.6%)	0.181 (-47.0%)	9.245 (-46.7%)
$\text{ACC}^{\text{F}_1}_{\text{SVM}}$	0.022 (-5.2%)	1.153 (+6.3%)	0.052 (-22.9%)	2.309 (-21.9%)	0.110 (-67.8%)	7.019 (-59.5%)
$\text{ACC}^{\text{AE}}_{\text{SVM}}$	0.020 (-11.4%)	0.933 (-13.9%)	0.069 (+1.6%)	3.193 (+7.9%)	0.108 (-68.4%)	7.225 (-58.4%)
$\text{ACC}^{\emptyset}_{\text{LR}}$	0.017	0.569	0.279	9.997	0.500	25.508
$\text{ACC}^{\text{A}}_{\text{LR}}$	0.020 (+21.2%)	0.933 (+63.9%)	0.060 (-78.6%)	2.628 (-73.7%)	0.185 (-62.9%)	9.629 (-62.3%)
$\text{ACC}^{\text{F}_1}_{\text{LR}}$	0.019 (+15.9%)	0.896 (+57.4%)	0.057 (-79.5%)	2.507 (-74.9%)	0.098 (-80.5%)	6.534 (-74.4%)
$\text{ACC}^{\text{AE}}_{\text{LR}}$	0.018 (+10.8%)	0.850 (+49.3%)	0.065 (-76.9%)	2.891 (-71.1%)	0.092 (-81.7%)	5.849 (-77.1%)
$\text{ACC}^{\emptyset}_{\text{RF}}$	0.034	1.254	0.136	4.199	0.439	23.528
$\text{ACC}^{\text{A}}_{\text{RF}}$	0.021 (-39.2%)	0.609 (-51.4%)	0.336 (+146.3%)	17.954 (+327.6%)	0.495 (+12.6%)	25.207 (+7.1%)
$\text{ACC}^{\text{F}_1}_{\text{RF}}$	0.021 (-39.2%)	0.594 (-52.6%)	0.153 (+12.5%)	6.095 (+45.2%)	0.490 (+11.5%)	24.996 (+6.2%)
$\text{ACC}^{\text{AE}}_{\text{RF}}$	0.020 (-39.5%)	0.622 (-50.4%)	0.159 (+16.3%)	5.996 (+42.8%)	0.495 (+12.8%)	25.382 (+7.9%)
$\text{ACC}^{\emptyset}_{\text{MNB}}$	0.049	2.316	0.473	23.280	0.500	25.508
$\text{ACC}^{\text{A}}_{\text{MNB}}$	0.051 (+4.4%)	2.479 (+7.0%)	0.189 (-59.9%)	9.065 (-61.1%)	0.435 (-13.1%)	22.170 (-13.1%)
$\text{ACC}^{\text{F}_1}_{\text{MNB}}$	0.049 (+0.5%)	2.404 (+3.8%)	0.197 (-58.3%)	9.285 (-60.1%)	0.428 (-14.5%)	22.025 (-13.7%)
$\text{ACC}^{\text{AE}}_{\text{MNB}}$	0.051 (+3.9%)	2.591 (+11.9%)	0.213 (-54.9%)	10.376 (-55.4%)	0.451 (-9.7%)	23.146 (-9.3%)
$\text{ACC}^{\emptyset}_{\text{CNN}}$	0.021	1.082	0.074	1.596	0.173	10.642
$\text{ACC}^{\text{A}}_{\text{CNN}}$	0.019 (-8.2%)	0.811 (-25.0%)	0.064 (-12.7%)	1.515 (-5.1%)	0.223 (+28.6%)	9.939 (-6.6%)
$\text{ACC}^{\text{F}_1}_{\text{CNN}}$	0.023 (+10.1%)	1.067 (-1.4%)	0.061 (-17.4%)	1.424 (-10.8%)	0.182 (+5.3%)	10.344 (-2.8%)
$\text{ACC}^{\text{AE}}_{\text{CNN}}$	0.023 (+9.1%)	1.072 (-0.9%)	0.068 (-7.8%)	1.399 (-12.4%)	0.174 (+0.7%)	10.810 (+1.6%)

Table 4. Same as Table 2, but with PCC instead of CC.

	IMDB		Kindle		HP	
	AE	RAE	AE	RAE	AE	RAE
$\text{PCC}^{\emptyset}_{\text{SVM}}$	0.101	9.460	0.255	14.514	0.375	20.158
$\text{PCC}^{\text{A}}_{\text{SVM}}$	0.100 (-0.4%)	9.517 (+0.6%)	0.283 (+10.9%)	16.174 (+11.4%)	0.385 (+2.6%)	20.653 (+2.5%)
$\text{PCC}^{\text{F}_1}_{\text{SVM}}$	0.101 (+0.0%)	9.425 (-0.4%)	0.251 (-1.8%)	14.239 (-1.9%)	0.385 (+2.7%)	20.594 (+2.2%)
$\text{PCC}^{\text{AE}}_{\text{SVM}}$	0.100 (-0.4%)	9.484 (+0.2%)	0.254 (-0.6%)	14.461 (-0.4%)	0.386 (+2.8%)	20.607 (+2.2%)
$\text{PCC}^{\emptyset}_{\text{LR}}$	0.122	11.564	0.356	20.405	0.464	24.608
$\text{PCC}^{\text{A}}_{\text{LR}}$	0.091 (-25.5%)	8.563 (-26.0%)	0.279 (-21.5%)	15.031 (-26.3%)	0.352 (-24.2%)	18.605 (-24.4%)
$\text{PCC}^{\text{F}_1}_{\text{LR}}$	0.092 (-25.0%)	8.606 (-25.6%)	0.172 (-51.6%)	11.222 (-45.0%)	0.212 (-54.2%)	16.117 (-34.5%)
$\text{PCC}^{\text{AE}}_{\text{LR}}$	0.079 (-35.3%)	7.348 (-36.5%)	0.154 (-56.6%)	13.066 (-36.0%)	0.211 (-54.6%)	19.597 (-20.4%)
$\text{PCC}^{\emptyset}_{\text{RF}}$	0.199	18.865	0.376	21.592	0.461	24.267
$\text{PCC}^{\text{A}}_{\text{RF}}$	0.214 (+7.6%)	20.313 (+7.7%)	0.388 (+3.3%)	22.011 (+1.9%)	0.470 (+2.0%)	24.722 (+1.9%)
$\text{PCC}^{\text{F}_1}_{\text{RF}}$	0.214 (+7.7%)	20.337 (+7.8%)	0.380 (+1.0%)	21.768 (+0.8%)	0.473 (+2.8%)	24.871 (+2.5%)
$\text{PCC}^{\text{AE}}_{\text{RF}}$	0.215 (+8.1%)	20.393 (+8.1%)	0.387 (+2.9%)	21.968 (+1.7%)	0.468 (+1.7%)	24.657 (+1.6%)
$\text{PCC}^{\emptyset}_{\text{MNB}}$	0.171	15.928	0.478	24.702	0.498	25.453
$\text{PCC}^{\text{A}}_{\text{MNB}}$	0.168 (-1.7%)	15.663 (-1.7%)	0.381 (-20.3%)	20.396 (-17.4%)	0.497 (-0.2%)	25.397 (-0.2%)
$\text{PCC}^{\text{F}_1}_{\text{MNB}}$	0.167 (-2.2%)	15.617 (-2.0%)	0.380 (-20.4%)	20.369 (-17.5%)	0.473 (-5.0%)	24.487 (-3.8%)
$\text{PCC}^{\text{AE}}_{\text{MNB}}$	0.160 (-6.4%)	14.907 (-6.4%)	0.380 (-20.4%)	20.396 (-17.4%)	0.473 (-5.0%)	24.479 (-3.8%)
$\text{PCC}^{\emptyset}_{\text{CNN}}$	0.110	9.994	0.111	10.448	0.257	18.368
$\text{PCC}^{\text{A}}_{\text{CNN}}$	0.105 (-4.8%)	9.893 (-1.0%)	0.154 (+39.2%)	10.775 (+3.1%)	0.389 (+51.6%)	21.093 (+14.8%)
$\text{PCC}^{\text{F}_1}_{\text{CNN}}$	0.099 (-10.3%)	9.377 (-6.2%)	0.111 (+0.3%)	9.474 (-9.3%)	0.251 (-2.2%)	17.005 (-7.4%)
$\text{PCC}^{\text{AE}}_{\text{CNN}}$	0.145 (+31.3%)	11.146 (+11.5%)	0.148 (+33.8%)	14.017 (+34.2%)	0.156 (-39.3%)	14.644 (-20.3%)

4.4 Results

Tables 2, 3, 4, and 5 report the results obtained for CC, ACC, PCC, and PACC. At a first glance, the results do not seem to give any clearcut indication on how the CC variants should be optimised. However, a closer look reveals a number of patterns. One of these is that SVM and LR (the two best-performing classifiers overall) tend to benefit from optimised hyperparameters, and tend to do so to a greater extent when the loss used in the optimisation is quantification-oriented. Somehow surprisingly, not all methods improve after model selection in every case. However, there tends to be such an improvement especially for ACC and PACC. A likely reason for this is the possible existence of a complex tradeoff between obtaining a more accurate classifier and obtaining more reliable estimates for the TPR and FPR quantities.

Regarding the different datasets, it seems that there is no clear improvement from performing model selection when the training set is balanced (see IMDB), neither by using a classification-oriented measure nor by using a quantification-oriented one. A possible reason is that any classifier (with or without hyperparameter op-

Table 5. Same as Table 2, but with PACC instead of CC.

	IMDB				Kindle				HP			
	AE		RAE		AE		RAE		AE		RAE	
$\text{PACC}^{\varnothing}_{\text{SVM}}$	0.021		1.166		0.059		2.464		0.137		8.368	
$\text{PACC}^{\text{A}}_{\text{SVM}}$	0.021	(-3.2%)	1.215	(+4.3%)	0.065	(+10.0%)	2.893	(+17.4%)	0.106	(-22.8%)	6.425	(-23.2%)
$\text{PACC}^{\text{F}_1}_{\text{SVM}}$	0.021	(-3.4%)	1.202	(+3.1%)	0.066	(+11.4%)	2.979	(+20.9%)	0.148	(+8.2%)	8.723	(+4.2%)
$\text{PACC}^{\text{AE}}_{\text{SVM}}$	0.022	(+5.1%)	1.363	(+17.0%)	0.059	(-1.4%)	2.333	(-5.3%)	0.114	(-16.6%)	7.497	(-10.4%)
$\text{PACC}^{\varnothing}_{\text{LR}}$	0.017		0.846		0.064		2.456		0.119		9.639	
$\text{PACC}^{\text{A}}_{\text{LR}}$	0.021	(+22.0%)	1.087	(+28.4%)	0.053	(-16.7%)	2.177	(-11.4%)	0.147	(+23.1%)	8.316	(-13.7%)
$\text{PACC}^{\text{F}_1}_{\text{LR}}$	0.021	(+24.5%)	1.176	(+39.0%)	0.065	(+2.2%)	2.060	(-16.1%)	0.091	(-23.2%)	7.748	(-19.6%)
$\text{PACC}^{\text{AE}}_{\text{LR}}$	0.021	(+26.5%)	1.237	(+46.3%)	0.068	(+5.5%)	2.253	(-8.3%)	0.104	(-12.3%)	8.812	(-8.6%)
$\text{PACC}^{\varnothing}_{\text{RF}}$	0.030		1.221		0.074		2.923		0.168		10.322	
$\text{PACC}^{\text{A}}_{\text{RF}}$	0.020	(-33.2%)	0.791	(-35.2%)	0.077	(+4.4%)	3.236	(+10.7%)	0.166	(-1.1%)	10.890	(+5.5%)
$\text{PACC}^{\text{F}_1}_{\text{RF}}$	0.019	(-35.8%)	0.833	(-31.8%)	0.070	(-5.1%)	2.169	(-25.8%)	0.176	(+4.9%)	11.507	(+11.5%)
$\text{PACC}^{\text{AE}}_{\text{RF}}$	0.020	(-32.9%)	0.826	(-32.3%)	0.064	(-14.1%)	2.390	(-18.2%)	0.122	(-27.6%)	8.584	(-16.8%)
$\text{PACC}^{\varnothing}_{\text{MNB}}$	0.055		3.253		0.180		7.352		0.195		10.930	
$\text{PACC}^{\text{A}}_{\text{MNB}}$	0.058	(+4.8%)	3.412	(+4.9%)	0.130	(-27.7%)	6.058	(-17.6%)	0.335	(+71.6%)	17.883	(+63.6%)
$\text{PACC}^{\text{F}_1}_{\text{MNB}}$	0.060	(+8.1%)	3.487	(+7.2%)	0.122	(-32.2%)	5.570	(-24.2%)	0.363	(+86.0%)	18.138	(+65.9%)
$\text{PACC}^{\text{AE}}_{\text{MNB}}$	0.063	(+14.9%)	3.815	(+17.3%)	0.144	(-19.6%)	6.626	(-9.9%)	0.248	(+27.2%)	13.999	(+28.1%)
$\text{PACC}^{\varnothing}_{\text{CNN}}$	0.022		1.205		0.064		1.414		0.181		9.808	
$\text{PACC}^{\text{A}}_{\text{CNN}}$	0.019	(-11.1%)	0.970	(-19.5%)	0.079	(+23.0%)	1.664	(+17.7%)	0.161	(-11.3%)	9.293	(-5.3%)
$\text{PACC}^{\text{F}_1}_{\text{CNN}}$	0.019	(-14.4%)	0.928	(-23.0%)	0.073	(+13.0%)	1.464	(+3.5%)	0.169	(-6.5%)	9.034	(-7.9%)
$\text{PACC}^{\text{AE}}_{\text{CNN}}$	0.018	(-17.3%)	0.830	(-31.2%)	0.069	(+6.9%)	1.367	(-3.3%)	0.165	(-9.1%)	8.829	(-10.0%)

Table 6. Two-sided t-test results on *related* sets of error scores across datasets and learners. For a comparison of two optimization measures X vs. Y, symbol $\gg$ (resp. $>$) indicates that method X performs better (i.e., yields lower error) than Y, and that the difference in performance, as averaged across pairs of experiments through datasets and learners, is statistically significant at a confidence score of $\alpha = 0.001$ (resp. $\alpha = 0.05$). Symbols $\ll$ and $<$ hold similar meaning but indicate X performs worse (i.e., yields higher error) than Y. Symbol $\sim$ instead indicates that the difference in performance between X and Y is not different in a statistically significant sense, i.e., that p-value >= 0.05.

	CC		ACC		PCC		PACC	
	AE	RAE	AE	RAE	AE	RAE	AE	RAE
AE vs. F$_1$	$\gg$	$\sim$	$\ll$	$\ll$	$\gg$	$\ll$	$\gg$	$\gg$
AE vs. A	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$
AE vs. Ø	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$
F$_1$ vs. A	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\ll$	$\sim$
F$_1$ vs. Ø	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\ll$	$\ll$
A vs. Ø	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\gg$	$\ll$	$\ll$

timisation) becomes a reasonable quantifier if it tends to pay equal importance to positive and negative examples, i.e., if the errors it produces are unbiased towards either $\oplus$ or $\ominus$. In this respect, RF and MNB prove strongly biased towards the majority class, and only when corrected via an adjustment (ACC or PACC) they deliver results comparable to those obtained for other learners.

CNN worked well on average almost in all cases, and seems to be the least sensitive learner to model selection.

In order to better understand whether or not, on average and across different situations, CC and its variants benefit from performing model selection using a quantification-oriented loss, we have submitted our results to a statistical significance test. Table 6 shows the outcome of a two-sided t-test on *related* sets of scores, across datasets and learners, from which we can compare pairs of model selection methods. The test reveals that optimising AE works better than optimising A or than using default settings (Ø). The test does not clearly say whether optimising AE or F$_1$ is better, but it suggests that PACC (the strongest CC variant) works better when optimised for AE than for F$_1$.

Table 7. Results showing how CC and its variants, once optimised using a quantification-oriented measure, compare with state-of-the-art more modern quantification methods. **Boldface** indicates the best method. All scores are different, in a statistically significant sense, from the best one according to a paired sample, two-tailed t-test at a confidence level of 0.001.

		IMDB		Kindle		HP	
		AE	RAE	AE	RAE	AE	RAE
CC and its variants	$\mathrm{CC}^{\mathrm{AE}}_{\mathrm{SVM}}$	0.065	6.091	0.100	7.555	0.119	10.593
	$\mathrm{ACC}^{\mathrm{AE}}_{\mathrm{SVM}}$	0.020	0.933	0.069	3.193	0.108	7.225
	$\mathrm{PCC}^{\mathrm{AE}}_{\mathrm{SVM}}$	0.100	9.484	0.254	14.461	0.386	20.607
	$\mathrm{PACC}^{\mathrm{AE}}_{\mathrm{SVM}}$	0.022	1.363	0.059	2.333	0.114	7.497
	$\mathrm{CC}^{\mathrm{AE}}_{\mathrm{LR}}$	0.062	5.745	0.094	7.087	0.110	10.304
	$\mathrm{ACC}^{\mathrm{AE}}_{\mathrm{LR}}$	0.018	0.850	0.065	2.891	0.092	5.849
	$\mathrm{PCC}^{\mathrm{AE}}_{\mathrm{LR}}$	0.079	7.348	0.154	13.066	0.211	19.597
	$\mathrm{PACC}^{\mathrm{AE}}_{\mathrm{LR}}$	0.021	1.237	0.068	2.253	0.104	8.812
	$\mathrm{CC}^{\mathrm{AE}}_{\mathrm{RF}}$	0.081	7.589	0.481	24.590	0.500	25.508
	$\mathrm{ACC}^{\mathrm{AE}}_{\mathrm{RF}}$	0.020	0.622	0.159	5.996	0.495	25.382
	$\mathrm{PCC}^{\mathrm{AE}}_{\mathrm{RF}}$	0.215	20.393	0.387	21.968	0.468	24.657
	$\mathrm{PACC}^{\mathrm{AE}}_{\mathrm{RF}}$	0.020	0.826	0.064	2.390	0.122	8.584
	$\mathrm{CC}^{\mathrm{AE}}_{\mathrm{MNB}}$	0.097	8.431	0.443	22.701	0.499	25.464
	$\mathrm{ACC}^{\mathrm{AE}}_{\mathrm{MNB}}$	0.051	2.591	0.213	10.376	0.451	23.146
	$\mathrm{PCC}^{\mathrm{AE}}_{\mathrm{MNB}}$	0.160	14.907	0.380	20.396	0.473	24.479
	$\mathrm{PACC}^{\mathrm{AE}}_{\mathrm{MNB}}$	0.063	3.815	0.144	6.626	0.248	13.999
	$\mathrm{CC}^{\mathrm{AE}}_{\mathrm{CNN}}$	0.074	6.613	0.109	8.591	0.343	19.008
	$\mathrm{ACC}^{\mathrm{AE}}_{\mathrm{CNN}}$	0.023	1.072	0.068	1.399	0.174	10.810
	$\mathrm{PCC}^{\mathrm{AE}}_{\mathrm{CNN}}$	0.145	11.146	0.148	14.017	0.156	14.644
	$\mathrm{PACC}^{\mathrm{AE}}_{\mathrm{CNN}}$	0.018	0.830	0.069	1.367	0.165	8.829
Other methods	$\mathrm{EMQ}^{\mathrm{AE}}_{\mathrm{LR}}$	**0.014**	**0.216**	**0.048**	1.606	**0.042**	**0.195**
	$\mathrm{SVM(KLD)}^{\mathrm{AE}}$	0.064	5.936	0.122	7.866	0.185	12.185
	$\mathrm{SVM(NKLD)}^{\mathrm{AE}}$	0.065	5.927	0.085	6.693	0.121	9.566
	$\mathrm{SVM(Q)}^{\mathrm{AE}}$	0.064	5.928	0.208	11.384	0.386	19.956
	$\mathrm{SVM(AE)}^{\mathrm{AE}}$	0.060	5.572	0.159	9.705	0.219	13.090
	$\mathrm{SVM(RAE)}^{\mathrm{RAE}}$	0.063	5.957	0.152	9.242	0.239	13.575
	$\mathrm{HDy}^{\mathrm{AE}}_{\mathrm{LR}}$	0.018	0.420	0.055	**1.027**	0.058	2.970
	$\mathrm{QuaNet}^{\mathrm{AE}}_{\mathrm{CNN}}$	0.027	1.175	0.070	2.119	0.210	11.433
	$\mathrm{MLPE}^{\varnothing}_{\varnothing}$	0.262	24.874	0.429	25.266	0.484	25.447

5 Conclusions

One of the takeaway messages from the present work is that, when using CC and/or its variants as baselines in their research on learning to quantify, researchers should properly optimise these baselines (i.e., use a true quantification loss in hyperparameter optimisation), lest these baselines become strawmen. The extensive empirical evaluation we have carried out shows that, in general, the performance of CC and its variants improves when the underlying learner has been optimised with a quantification-oriented loss (AE). The results of our experiments are less clear about whether optimising AE of F_1 (which, despite being a *classification*-oriented loss, is one that rewards classifiers that balance FPs and FNs) is better, although they indicate that optimising AE is preferable for PACC, the strongest among the variants of CC.

Acknowledgments

The present work has been supported by the SoBigdata++ project, funded by the European Commission (Grant 871042) under the H2020 Programme INFRAIA-2019-1, and by the AI4Media project, funded by the European Commission (Grant 951911) under the H2020 Programme ICT-48-2020. The authors' opinions do not necessarily reflect those of the European Commission.

References

1. Barranquero, J., Díez, J., del Coz, J.J.: Quantification-oriented learning based on reliable classifiers. Pattern Recognition **48**(2), 591–604 (2015). https://doi.org/10.1016/j.patcog.2014.07.032
2. Barranquero, J., González, P., Díez, J., del Coz, J.J.: On the study of nearest neighbor algorithms for prevalence estimation in binary problems. Pattern Recognition **46**(2), 472–482 (2013). https://doi.org/10.1016/j.patcog.2012.07.022
3. Bella, A., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Quantification via probability estimators. In: Proceedings of the 11th IEEE International Conference on Data Mining (ICDM 2010). pp. 737–742. Sydney, AU (2010). https://doi.org/10.1109/icdm.2010.75
4. Borge-Holthoefer, J., Magdy, W., Darwish, K., Weber, I.: Content and network dynamics behind Egyptian political polarization on Twitter. In: Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW 2015). pp. 700–711. Vancouver, CA (2015)
5. Card, D., Smith, N.A.: 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). pp. 1636–1646. New Orleans, US (2018). https://doi.org/10.18653/v1/n18-1148
6. Esuli, A., Moreo, A., Sebastiani, F.: A recurrent neural network for sentiment quantification. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM 2018). pp. 1775–1778. Torino, IT (2018). https://doi.org/10.1145/3269206.3269287
7. Esuli, A., Moreo, A., Sebastiani, F.: Cross-lingual sentiment quantification. IEEE Intelligent Systems **35**(3), 106–114 (2020). https://doi.org/10.1109/MIS.2020.2979203
8. Esuli, A., Sebastiani, F.: Explicit loss minimization in quantification applications (preliminary draft). In: Proceedings of the 8th International Workshop on Information Filtering and Retrieval (DART 2014). pp. 1–11. Pisa, IT (2014)
9. Esuli, A., Sebastiani, F.: Optimizing text quantifiers for multivariate loss functions. ACM Transactions on Knowledge Discovery and Data **9**(4), Article 27 (2015). https://doi.org/10.1145/2700406
10. Forman, G.: Quantifying counts and costs via classification. Data Mining and Knowledge Discovery **17**(2), 164–206 (2008). https://doi.org/10.1007/s10618-008-0097-y
11. Gao, W., Sebastiani, F.: From classification to quantification in tweet sentiment analysis. Social Network Analysis and Mining **6**(19), 1–22 (2016). https://doi.org/10.1007/s13278-016-0327-z
12. González, P., Castaño, A., Chawla, N.V., del Coz, J.J.: A review on quantification learning. ACM Computing Surveys **50**(5), 74:1–74:40 (2017). https://doi.org/10.1145/3117807
13. González, P., Díez, J., Chawla, N., del Coz, J.J.: Why is quantification an interesting learning problem? Progress in Artificial Intelligence **6**(1), 53–58 (2017). https://doi.org/10.1007/s13748-016-0103-3
14. González-Castro, V., Alaiz-Rodríguez, R., Alegre, E.: Class distribution estimation based on the Hellinger distance. Information Sciences **218**, 146–164 (2013). https://doi.org/10.1016/j.ins.2012.05.028
15. Hassan, W., Maletzke, A., Batista, G.: Accurately quantifying a billion instances per second. In: Proceedings of the 7th IEEE International Conference on Data Science and Advanced Analytics (DSAA 2020). Sydney, AU (2020)
16. Hopkins, D.J., King, G.: A method of automated nonparametric content analysis for social science. American Journal of Political Science **54**(1), 229–247 (2010). https://doi.org/10.1111/j.1540-5907.2009.00428.x
17. Joachims, T.: A support vector method for multivariate performance measures. In: Proceedings of the 22nd International Conference on Machine Learning (ICML 2005). pp. 377–384. Bonn, DE (2005)
18. Levin, R., Roitman, H.: Enhanced probabilistic classify and count methods for multi-label text quantification. In: Proceedings of the 7th ACM International Conference on the Theory of Information Retrieval (ICTIR 2017). pp. 229–232. Amsterdam, NL (2017). https://doi.org/10.1145/3121050.3121083

19. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011). pp. 142–150. Portland, US (2011)
20. Milli, L., Monreale, A., Rossetti, G., Giannotti, F., Pedreschi, D., Sebastiani, F.: Quantification trees. In: Proceedings of the 13th IEEE International Conference on Data Mining (ICDM 2013). pp. 528–536. Dallas, US (2013). https://doi.org/10.1109/icdm.2013.122
21. Moreno-Torres, J.G., Raeder, T., Alaíz-Rodríguez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognition **45**(1), 521–530 (2012). https://doi.org/10.1016/j.patcog.2011.06.019
22. Morik, K., Brockhausen, P., Joachims, T.: Combining statistical learning with a knowledge-based approach. A case study in intensive care monitoring. In: Proceedings of the 16th International Conference on Machine Learning (ICML 1999). pp. 268–277. Bled, SL (1999)
23. Platt, J.C.: Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In: Smola, A., Bartlett, P., Schölkopf, B., Schuurmans, D. (eds.) Advances in Large Margin Classifiers, pp. 61–74. The MIT Press, Cambridge, MA (2000)
24. Pérez-Gállego, P., Castaño, A., Quevedo, J.R., del Coz, J.J.: Dynamic ensemble selection for quantification tasks. Information Fusion **45**, 1–15 (2019). https://doi.org/10.1016/j.inffus.2018.01.001
25. Pérez-Gállego, P., Quevedo, J.R., del Coz, J.J.: Using ensembles for problems with characterizable changes in data distribution: A case study on quantification. Information Fusion **34**, 87–100 (2017). https://doi.org/10.1016/j.inffus.2016.07.001
26. Saerens, M., Latinne, P., Decaestecker, C.: Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. Neural Computation **14**(1), 21–41 (2002). https://doi.org/10.1162/089976602753284446
27. Sebastiani, F.: Evaluation measures for quantification: An axiomatic approach. Information Retrieval Journal **23**(3), 255–288 (2020). https://doi.org/10.1007/s10791-019-09363-y