Optimizing Text Quantifiers for Multivariate Loss Functions

ANDREA ESULI, Consiglio Nazionale delle Ricerche
FABRIZIO SEBASTIANI, Qatar Computing Research Institute

We address the problem of *quantification*, a supervised learning task whose goal is, given a class, to estimate the relative frequency (or *prevalence*) of the class in a dataset of unlabeled items. Quantification has several applications in data and text mining, such as estimating the prevalence of positive reviews in a set of reviews of a given product or estimating the prevalence of a given support issue in a dataset of transcripts of phone calls to tech support. So far, quantification has been addressed by learning a general-purpose classifier, counting the unlabeled items that have been assigned the class, and tuning the obtained counts according to some heuristics. In this article, we depart from the tradition of using general-purpose classifiers and use instead a supervised learning model for *structured prediction*, capable of generating classifiers directly optimized for the (multivariate and nonlinear) function used for evaluating quantification accuracy. The experiments that we have run on 5,500 binary high-dimensional datasets (averaging more than 14,000 documents each) show that this method is more accurate, more stable, and more efficient than existing state-of-the-art quantification methods.

Categories and Subject Descriptors: I.5.2 [**Pattern Recognition**]: Design Methodology—*Classifier design and evaluation*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering; Search process*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

General Terms: Algorithm, Design, Experimentation, Measurements

Additional Key Words and Phrases: Quantification, prevalence estimation, prior estimation, supervised learning, text classification, loss functions, Kullback-Leibler divergence

ACM Reference Format:
Andrea Esuli and Fabrizio Sebastiani. 2015. Optimizing text quantifiers for multivariate loss functions. ACM Trans. Knowl. Discov. Data 9, 4, Article 27 (June 2015), 27 pages.
DOI: http://dx.doi.org/10.1145/2700406

1. INTRODUCTION

In recent years, it has been pointed out that in a number of applications involving classification, the final goal is not determining which class (or classes) individual unlabeled data items belong to, but determining the *prevalence* (or "relative frequency") of each class in the unlabeled data. The latter task is known as *quantification* [Forman 2005, 2006a, 2008; Forman et al. 2006].

Although what we are going to discuss here applies to any type of data, we are mostly interested in *text* quantification, that is, quantification when the data items are textual documents. To see the importance of text quantification, let us examine the

Authors' addresses: A. Esuli, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Via Giuseppe Moruzzi 1, 56124 Pisa, Italy; email: andrea.esuli@isti.cnr.it. F. Sebastiani, Qatar Computing Research Institute, PO Box 5825, Doha, Qatar; email: fsebastiani@qf.org.qa. F. Sebastiani is on leave from Consiglio Nazionale delle Ricerche. The order in which the authors are listed is purely alphabetical; each author has given an equally important contribution to this work.

DOI: http://dx.doi.org/10.1145/2700406

task of classifying textual answers returned to open-ended questions in questionnaires [Esuli and Sebastiani 2010a; Gamon 2004; Giorgetti and Sebastiani 2003], and let us discuss two important such scenarios.

In the first scenario, a telecommunications company asks its current customers the question "How satisfied are you with our mobile phone services?" and wants to classify the resulting textual answers according to whether they belong to the class MayDefectToCompetition. The company is likely interested in accurately classifying each individual customer, since it may want to call each customer that is assigned the class and offer him or her improved conditions.

In the second scenario, a market research agency asks respondents the question "What do you think of the recent ad campaign for product X?" and wants to classify the resulting textual answers according to whether they belong to the class LovedTheCampaign. Here, the agency is likely *not* interested in whether a specific individual belongs to the class LovedTheCampaign but is likely interested in knowing *how many* respondents belong to it, that is, in knowing the prevalence of the class.

In sum, while in the first scenario classification is the goal, in the second scenario the real goal is quantification, that is, evaluating the results of classification at the *aggregate* level rather than at the *individual* level. Other scenarios in which quantification is the goal may be, for example, predicting election results by estimating the prevalence of blog posts (or tweets) supporting a given candidate or party [Hopkins and King 2010], planning the amount of human resources to allocate to different types of issues in a customer support center by estimating the prevalence of customer calls related to a given issue [Forman 2005], supporting epidemiological research by estimating the prevalence of medical reports in which a specific pathology is diagnosed [Baccianella et al. 2013].

The obvious method for dealing with the latter type of scenario is *aggregative quantification*, that is, classifying each unlabeled document and estimating class prevalence by counting the documents that have been attributed the class. However, there are two reasons this strategy is suboptimal. The first reason is that a good classifier may not be a good quantifier, and vice versa. To see this, one only needs to look at the definition of F_1, the standard evaluation function for binary classification, defined as

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}, \tag{1}$$

where TP, FP, and FN indicate the numbers of true positives, false positives, and false negatives, respectively. According to F_1, a binary classifier $\hat{\Phi}_1$ for which $FP = 20$ and $FN = 20$ is worse than a classifier $\hat{\Phi}_2$ for which, on the same test set, $FP = 0$ and $FN = 10$. However, $\hat{\Phi}_1$ is intuitively a better binary quantifier than $\hat{\Phi}_2$; indeed, $\hat{\Phi}_1$ is a perfect quantifier, since FP and FN are equal and thus compensate each other, so that the distribution of the test items across the class and its complement is estimated perfectly.

A second reason is that standard supervised learning algorithms are based on the assumption that the training set is drawn from the same distribution as the unlabeled data the classifier is supposed to classify. But in real-world settings, this assumption is often violated, a phenomenon usually referred to as *concept drift* [Sammut and Harries 2011]. For instance, in a backlog of newswire stories from 2001, the prevalence of the class Terrorism in August data will likely not be the same as in September data; training on August data and testing on September data might well yield low quantification accuracy. Violations of this assumption may occur "for reasons ranging from the bias introduced by experimental design, to the irreproducibility of the testing conditions at training time" [Quiñonero-Candela et al. 2009]. Concept drift usually comes in one of three forms [Kelly et al. 1999]: (1) the class priors $p(c_i)$ may change (i.e., the one

in the test set may significantly differ from the one in the training set); (2) the class-conditional distributions $p(\mathbf{x}|c_i)$ may change; or (c) the posterior distribution $p(c_i|\mathbf{x})$ may change. It is the first of these three cases that poses a problem for quantification.

The previous arguments indicate that text quantification should not be considered a mere byproduct of text classification and should be studied as a task of its own. To date, proposed methods explicitly addressed to quantification (see, e.g., Bella et al. [2010], Forman [2005, 2006a, 2008], Forman et al. [2006], Hopkins and King [2010], and Xue and Weiss [2009]) employ general-purpose supervised learning methods, that is, address quantification by elaborating on the results returned by a general-purpose standard classifier. In this article, we take a sharply different, *structured prediction* approach, based on the use of classifiers explicitly optimized for the nonlinear, multivariate evaluation function that we will use for assessing quantification accuracy. This idea was first proposed, but not implemented, by Esuli and Sebastiani [2010b].

The rest of the article is organized as follows. In Section 2, after setting the stage, we describe the evaluation function we will adopt (Section 2.1) and sketch a number of quantification methods previously proposed in the literature (Section 2.2). In Section 3, we introduce our novel method based on explicitly minimizing, via a structured prediction model, the evaluation measure we have chosen.

Section 4 presents experiments in which we test the method we have proposed on 5,148 binary, high-dimensional datasets (averaging more than 15,000 documents each and characterized by more than 50,000 features), using all the methods introduced in Section 2.2 as baselines. Section 4 presents experiments in which we test the method we propose on two large batches of binary, high-dimensional, publicly available datasets (the two batches consist of 5,148 and 352 datasets, respectively), using all the methods introduced in Section 2.2 as baselines. Section 5 discusses related work, while Section 6 concludes.

2. PRELIMINARIES

In this article, we will focus on quantification at the binary level. That is, given a domain of documents D and a class c, we assume the existence of an unknown *target function* (or *ground truth*) $\Phi : D \rightarrow \{-1, +1\}$ that specifies which members of D belong to c; as usual, $+1$ and -1 represent membership and nonmembership in c, respectively. The approaches we will focus on are based on aggregative quantification; that is, they rely on the generation of a classifier $\hat{\Phi} : D \rightarrow \{-1, +1\}$ via supervised learning from a training set Tr. We will indicate with Te the test set on which quantification effectiveness is going to be tested.

We define the *prevalence* (or *relative frequency*) $\lambda_{Te}(c)$ of class c in a set of documents Te as the fraction of members of Te that belong to c, that is, as

$$\lambda_{Te}(c) = \frac{|\{d_j \in Te | \Phi(d_j) = +1\}|}{|Te|}. \tag{2}$$

Given a set Te of unlabeled documents and a class c, quantification is defined as the task of estimating $\lambda_{Te}(c)$, that is, of computing an estimate $\hat{\lambda}_{Te}(c)$ such that $\lambda_{Te}(c)$ and $\hat{\lambda}_{Te}(c)$ are as close as possible.[1] What "as close as possible" exactly means will be formalized by an appropriate evaluation measure (see Section 2.1).

The reasons we focus on *binary* quantification are twofold:

—Many quantification problems are binary in nature. For instance, estimating the prevalence of positive and negative reviews in a dataset of reviews of a given product is such a task. Another such task is estimating from blog posts the prevalence of

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

support for either of two candidates in the second round of a two-round ("run-off") election.

—A multiclass multilabel problem (also known as an *n-of-m problem*, that is, a problem where zero, one, or several among m classes can be attributed to the same document) can be reduced to m independent binary problems of type (c_j vs. $\overline{c}_j$), where $\mathcal{C} = \{c_1, \ldots, c_j, \ldots, c_m\}$ is the set of classes and where $\overline{c}_j$ denotes the complement of c_j. Binary quantification methods can thus also be applied to solving quantification in multiclass multilabel contexts.

We instead leave the discussion of quantification in single-label multiclass (i.e., 1-of-m) contexts to future work.

2.1. Evaluation Measures for Quantification

Different measures have been used in the literature for measuring binary quantification accuracy.

The simplest such measure is *bias* (B), defined as $B(\lambda_{Te}, \hat{\lambda}_{Te}) = \hat{\lambda}_{Te}(c) - \lambda_{Te}(c)$ and used in Forman [2005, 2006a] and Tang et al. [2010]; positive bias indicates a tendency to overestimate the prevalence of c, while negative bias indicates a tendency to underestimate it.

Absolute Error (AE, also used by Esuli and Sebastiani [2010b], where it is called *percentage discrepancy*, and by Barranquero et al. [2013], Bella et al. [2010], Forman [2005, 2006a], González-Castro et al. [2013], Sánchez et al. [2008], and Tang et al. [2010]), defined as $AE(\lambda_{Te}, \hat{\lambda}_{Te}) = |\hat{\lambda}_{Te}(c) - \lambda_{Te}(c)|$, is an alternative, equally simplistic measure that accounts for the fact that positive and negative bias are (in the absence of specific application-dependent constraints) equally undesirable.

Relative absolute error (RAE), defined as

$$RAE(\lambda_{Te}, \hat{\lambda}_{Te}) = \frac{|\hat{\lambda}_{Te}(c) - \lambda_{Te}(c)|}{\lambda_{Te}(c)}, \tag{3}$$

is a refinement of AE meant to account for the fact that the same value of absolute error is a more serious mistake when the true class prevalence is small. For instance, predicting $\hat{\lambda}_{Te}(c) = 0.10$ when $\lambda_{Te}(c) = 0.01$ and predicting $\hat{\lambda}_{Te}(c) = 0.50$ when $\lambda_{Te}(c) = 0.41$ are equivalent errors according to B and AE, but the former is intuitively a more serious error than the latter.

The most convincing among the evaluation measures proposed so far is certainly Forman's [2005], who uses *normalized cross-entropy*, better known as *Kullback-Leibler Divergence* (KLD—see e.g., Cover and Thomas [1991]). KLD, defined as

$$KLD(\lambda_{Te}, \hat{\lambda}_{Te}) = \sum_{c \in C} \lambda_{Te}(c) \log \frac{\lambda_{Te}(c)}{\hat{\lambda}_{Te}(c)} \tag{4}$$

and also used by Esuli and Sebastiani [2010b], Forman [2006a, 2008], and Tang et al. [2010], is a measure of the error made in estimating a true distribution λ_{Te} over a set C of classes by means of a distribution $\hat{\lambda}_{Te}$; this means that KLD is in principle suitable for evaluating quantification, since quantifying exactly means predicting how the test items are distributed across the classes. KLD ranges between 0 (perfect coincidence of λ_{Te} and $\hat{\lambda}_{Te}$) and $+\infty$ (total divergence of λ_{Te} and $\hat{\lambda}_{Te}$). In the binary case in which $C = \{c, \overline{c}\}$, KLD becomes

$$KLD(\lambda_{Te}, \hat{\lambda}_{Te}) = \lambda_{Te}(c) \log \frac{\lambda_{Te}(c)}{\hat{\lambda}_{Te}(c)} + \lambda_{Te}(\overline{c}) \log \frac{\lambda_{Te}(\overline{c})}{\hat{\lambda}_{Te}(\overline{c})}. \tag{5}$$

Continuity arguments indicate that we should consider $0 \log \frac{0}{q} = 0$ and $p \log \frac{p}{0} = +\infty$ (see Cover and Thomas [1991, p. 18]). Note that, as from Equation (4), KLD is undefined when the predicted distribution $\hat{\lambda}_{Te}$ is zero for at least one class (a problem that also affects RAE). As a result, we smooth the fractions $\lambda_{Te}(c)/\hat{\lambda}_{Te}(c)$ and $\lambda_{Te}(\overline{c})/\hat{\lambda}_{Te}(\overline{c})$ in Equation (4) by adding a small quantity ϵ to both the numerator and the denominator. The smoothed KLD function is always defined and still returns a value of zero when λ_{Te} and $\hat{\lambda}_{Te}$ coincide.

KLD offers several advantages with respect to RAE (and, a fortiori, to B and AE). One advantage is that, as evident from Equation (5), it is symmetric with respect to the complement of a class; that is, switching the role of c and $\overline{c}$ does not change the result. This means that, for example, predicting $\hat{\lambda}_{Te}(c) = 0.10$ when $\lambda_{Te}(c) = 0.11$ and predicting $\hat{\lambda}_{Te}(c) = 0.90$ when $\lambda_{Te}(c) = 0.89$ are equivalent errors (which seems intuitive), while RAE considers the former a much more serious error than the latter. This is especially useful in binary quantification tasks in which it is not clear which of the two classes should play the role of the positive class c, as in, for example, Employed vs. Unemployed. A second advantage is that KLD is not defined only on the binary (and multilabel multiclass) case, but is also defined on the single-label multiclass case; this allows evaluating different types of quantification tasks with the same measure. Last but not least, one benefit of using KLD is that it is a very well-known measure, having been the subject of intense study within information theory [Csiszár and Shields 2004] and, although from a more applicative angle, within the language modeling approach to information retrieval [Zhai 2008].

2.2. Existing Quantification Methods

A number of methods have been proposed in the (still brief) literature on quantification; here we list the main ones, which we will use as baselines in the experiments discussed in Section 4.

Classify and Count (CC). An obvious method for quantification consists of generating a classifier from Tr, classifying the documents in Te, and estimating λ_{Te} by simply counting the fraction of documents in Te that are predicted positive, that is,

$$\hat{\lambda}^{CC}_{Te}(c) = \frac{|\{d_j \in Te | \hat{\Phi}(d_j) = +1\}|}{|Te|}. \tag{6}$$

Forman [2008] calls this the *classify and count* (CC) method.

Probabilistic Classify and Count (PCC). A variant of the previous consists of generating a classifier from Tr, classifying the documents in Te, and computing λ_{Te} as the *expected* fraction of documents predicted positive, that is,

$$\hat{\lambda}^{PCC}_{Te}(c) = \frac{1}{|Te|} \sum_{d_j \in Te} p(c|d_j), \tag{7}$$

where $p(c|d_j)$ is the probability of membership in c of test document d_j returned by the classifier. If the classifier only returns confidence scores that are not probabilities (as is the case, e.g., when AdaBoost.MH is the learner [Schapire and Singer 2000]), the confidence scores must be converted into probabilities, for example, by applying a logistic function. The PCC method is dismissed as unsuitable by Forman [2005, 2008] but is shown to perform better than CC by Bella et al. [2010] (where it is called "Probability Average") and by Tang et al. [2010].

Adjusted Classify and Count (ACC). Forman [2005, 2008] uses a further method that he calls "Adjusted Count" and that we will call *Adjusted Classify and Count* (ACC) to make its relation with CC more explicit. The underlying idea is that CC would be

optimal were it not for the fact that the classifier may generate different numbers of false positives and false negatives, and that this difference would lead to imperfect quantification. If we knew the "true positive rate" ($tpr = \frac{TP}{TP+FN}$, a.k.a. recall) and "false positive rate" ($fpr = \frac{FP}{FP+TN}$, a.k.a. fallout) that the classifier has obtained on Te, it is easy to check that perfect quantification would be obtained by adjusting $\hat{\lambda}^{CC}_{Te}(c)$ as follows:

$$\hat{\lambda}^{ACC}_{Te}(c) = \frac{\hat{\lambda}^{CC}_{Te}(c) - fpr_{Te}(c)}{tpr_{Te}(c) - fpr_{Te}(c)}. \tag{8}$$

Since we cannot know the true values of $tpr_{Te}(c)$ and $fpr_{Te}(c)$, the ACC method consists of estimating them on Tr via k-fold cross-validation and using the resulting estimates in Equation (8).

However, one problem with ACC is that it is not guaranteed to return a value in [0,1], due to the fact that the estimates of $tpr_{Te}(c)$ and $fpr_{Te}(c)$ may be imperfect. This led Forman [2008] to "clip" the results of the estimation (i.e., equate to 1 every value higher than 1 and to 0 every value lower than 0) in order for the final results to be in [0,1].

Probabilistic Adjusted Classify and Count (PACC). The PACC method (proposed in Bella et al. [2010], where it is called "Scaled Probability Average") is a probabilistic variant of ACC; that is, it stands to ACC like PCC stands to CC. Its underlying idea is to replace, in Equation (8), $\hat{\lambda}^{CC}_{Te}(c)$, $tpr_{Te}(c)$, and $fpr_{Te}(c)$ with their expected values, with probability of membership in c replacing binary predictions. Equation (8) is thus transformed into

$$\hat{\lambda}^{PACC}_{Te}(c) = \frac{\hat{\lambda}^{PCC}_{Te}(c) - E[fpr_{Te}(c)]}{E[tpr_{Te}(c)] - E[fpr_{Te}(c)]}, \tag{9}$$

where $E[tpr_{Te}(c)]$ and $E[fpr_{Te}(c)]$ (*expected*$tpr_{Te}(c)$ and *expected* $fpr_{Te}(c)$, respectively) are defined as

$$E[tpr_{Te}(c)] = \frac{1}{|Te_c|} \sum_{d_j \in Te_c} p(c|d_j) \tag{10}$$

$$E[fpr_{Te}(c)] = \frac{1}{|Te_{\bar{c}}|} \sum_{d_j \in Te_{\bar{c}}} p(c|d_j), \tag{11}$$

and Te_c ($Te_{\bar{c}}$, respectively) indicates the set of documents in Te that belong (do not belong, respectively) to class c. Again, since we cannot know the true $E[tpr_{Te}(c)]$ and $E[fpr_{Te}(c)]$ (given that we do not know Te_c and $Te_{\bar{c}}$), we estimate them on Tr via k-fold cross-validation and use the resulting estimates in Equation (9).

Threshold@0.50 (T50), Method X (X), and Method Max (MAX). Forman [2008] points out that the ACC method is very sensitive to the decision threshold of the classifier, which may yield unreliable values of $\lambda^{ACC}_{Te}(c)$ (or lead to $\lambda^{ACC}_{Te}(c)$ being undefined when $tpr_{Te} = fpr_{Te}$). In order to reduce this sensitivity, Forman [2008] recommends heuristically setting the decision threshold in such a way that tpr_{Tr} (as obtained via k-fold cross-validation) is equal to .50 before computing Equation (8). This method is dubbed *Threshold@0.50* (T50). Alternative heuristics that Forman [2008] discusses are to set the decision threshold in such a way that $fpr_{Tr} = 1 - tpr_{Tr}$ (this is dubbed *Method X*) or such that $(tpr_{Tr} - fpr_{Tr})$ is maximized (this is dubbed *Method Max*).

Median Sweep (MS). Alternatively, Forman [2008] recommends computing $\lambda^{ACC}_{Te}(c)$ for every decision threshold that gives rise (in k-fold cross-validation) to different tpr_{Tr}

or fpr_{Tr} values and take the median of all the resulting estimates of $\lambda_{Te}^{ACC}(c)$. This method is dubbed *Median Sweep* (MS).

Mixture Model (MM). The MM method (proposed in Forman [2005]) consists of assuming that the distribution D^{Te} of the scores that the classifier assigns to the test examples is a mixture

$$D^{Te} = \lambda_{Te}(c) \cdot D_c^{Te} + (1 - \lambda_{Te}(c)) \cdot D_{\bar{c}}^{Te}, \tag{12}$$

where D_c^{Te} and $D_{\bar{c}}^{Te}$ are the distributions of the scores that the classifier assigns to the positive and the negative test examples, respectively, and where $\lambda_{Te}(c)$ and $\lambda_{Te}(\bar{c})$ are the parameters of this mixture. The MM method consists of estimating D_c^{Te} and $D_{\bar{c}}^{Te}$ via k-fold cross-validation and picking as value of $\lambda_{Te}(c)$ the one that generates the best fit between the observed D^{Te} and the mixture. Two variants of this method, called the *Kolmogorov-Smirnov Mixture Model* (MM(KS)) and the *PP-Area Mixture Model* (MM(PP)), are actually defined in Forman [2005], which differ in terms of how the goodness of fit between the left- and the right-hand side of Equation (12) is estimated. See Forman [2005] for more details.

3. OPTIMIZING QUANTIFICATION ACCURACY

A problem with the methods discussed in Section 2.2 is that most of them are fairly heuristic in nature. For instance, the fact that methods such as ACC (and all the others based on it, such as T50, MS, X, and MAX) require "clipping" is scarcely reassuring. More in general, methods such as T50 or MS have hardly any theoretical foundation, and choosing them over CC only rests on our knowledge that they have performed better in previously reported experiments.

A further problem is that some of these methods rest on assumptions that seem problematic. For instance, one problem with the MM method is that it seems to implicitly rely on the hypothesis that estimating D_c^{Te} and $D_{\bar{c}}^{Te}$ via k-fold cross-validation on Tr can be done reliably. However, since the very motivation of doing quantification is that the training set and the test set may have quite different characteristics, this hypothesis seems adventurous. A similar argument casts some doubt on ACC: how reliable are the estimates of tpr_{Te} and fpr_{Te} that can be generated via k-fold cross-validation on Tr, given the different characteristics that the training set and test set may have in the application contexts where quantification is required?[2] In sum, the very same arguments that are used to deem the CC method unsuitable for quantification seem to undermine the previously mentioned attempts at improving on CC.

In this article, we propose a new, theoretically well-founded quantification method that radically differs from the ones discussed in Section 2.2. Note that all of the methods discussed in Section 2.2 employ *general-purpose* supervised learning methods, that is, address quantification by postprocessing the results returned by a *standard* classifier (where the decision threshold has possibly been tuned according to some heuristics). In particular, all the supervised learning methods adopted in the literature on quantification optimize Hamming distance or variants thereof, and not a quantification-specific evaluation function. When the dataset is imbalanced (typically: when the positives are by far outnumbered by the negatives), as is frequently the case in text classification, this is suboptimal, since a supervised learning method that minimizes Hamming distance

[2]In Appendix A, we thoroughly analyze (also by means of concrete experiments) the issue of how (un)reliable the k-fold cross-validation estimates of tpr_{Te} and fpr_{Te} are in practice.

will generate classifiers with a tendency to make negative predictions. This means that FN will be much higher than FP, to the detriment of quantification accuracy.[3]

We take a sharply different approach based on the use of classifiers explicitly optimized for the evaluation function that we will use for assessing quantification accuracy. Given such a classifier, we will simply use a "classify and count" approach, with no heuristic threshold tuning (à la T50 / X / MAX) and no a posteriori adjustment (à la ACC).

The idea of using learning algorithms capable of directly optimizing the measure (a.k.a. "loss") used for evaluating effectiveness is well established in supervised learning. However, in our case, following this route is nontrivial, because the evaluation measure that we want to use (KLD) is nonlinear, that is, is such that the error on the test set may not be formulated as a linear combination of the error incurred by each test example. An evaluation measure for quantification is *inherently* nonlinear, because how the error on an individual test item impacts on the overall quantification error depends on how the other test items have been classified. For instance, if in the other test items there are more false positives than false negatives, an additional false negative is actually *beneficial* to overall quantification error, because of the mutual compensation effect between FP and FN mentioned in Section 1. As a result, a measure of quantification accuracy is inherently nonlinear and should thus be multivariate, that is, take in consideration all test items at once.

As discussed by Joachims [2005], the assumption that the error on the test set may be formulated as a linear combination of the error incurred by each test example (as indeed happens for many common error measures, e.g., Hamming distance) underlies most existing discriminative learners, which are thus suboptimal for tackling quantification. In order to sidestep this problem, we adopt the *SVM for Multivariate Performance Measures* (SVM_{perf}) learning algorithm proposed by Joachims [2005].[4] SVM_{perf} is a learning algorithm of the Support Vector Machine family that can generate classifiers optimized for any nonlinear, multivariate loss function that can be computed from a contingency table (as KLD is).

SVM_{perf} is a specialization to the problem of binary classification of the *structural SVM* (SVM^{struct}) learning algorithm [Joachims et al. 2009a, 2009b; Tsochantaridis et al. 2004] for "structured prediction," that is, an algorithm designed for predicting multivariate, structured objects (e.g., trees, sequences, sets). SVM_{perf} is fundamentally different from conventional algorithms for learning classifiers: while these latter learn univariate classifiers (i.e., functions of type $\hat{\Phi} : D \rightarrow \{-1, +1\}$ that classify individual instances one at a time), SVM_{perf} learns *multivariate* classifiers (i.e., functions of type $\hat{\Phi} : D^{|S|} \rightarrow \{-1, +1\}^{|S|}$ that classify *entire sets* S of instances in one shot). By doing so, SVM_{perf} can optimize properties of entire sets of instances, properties (such as KLD) that cannot be expressed as linear functions of the properties of the individual instances.

As discussed by Joachims et al. [2009b], SVM^{struct} can be adapted to a specific task by defining four components:

(1) A *joint feature map* $\Psi(\mathbf{x}, \mathbf{y})$. This function computes a vector of features (describing the match between the input vectors in $\mathbf{x}$ and the relative outputs, true or predicted,

[3]To witness, in the experiments we report in Section 4, our 5,148 test sets exhibit, when classified by the classifiers generated by the linear SVM used for implementing the CC method, an average FP/FN ratio of 0.109; by contrast, for an optimal quantifier, this ratio is always 1.

[4]In Joachims [2005], SVM_{perf} is actually called $SVM\Delta_{multi}$, but the author has released its implementation under the name SVM_{perf}. We will use this latter name because it uniquely identifies the algorithm on the web, while searching for "SVM multi" often returns the $SVM^{multiclass}$ package, which addresses a different problem.

in $\mathbf{y}$) from all the input–output pairs at the same time. In this way, the number of features, and thus the number of parameters of the model, can be kept constant regardless of the size of the sample set. The Ψ function allows one to generalize not only on inputs ($\mathbf{x}$) but also on outputs ($\mathbf{y}$), thus allowing one to produce predictions not seen in the training data.
In SVM$_{perf}$, Ψ is defined[5] as

$$\Psi(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} y_i \mathbf{x_i}. \tag{13}$$

(2) A loss function $\Delta(\mathbf{y}, \hat{\mathbf{y}})$. SVM$_{perf}$ works with loss functions $\Delta(TP, FP, FN, TN)$ in which the four values are those from the contingency table resulting from comparing the true labels $\mathbf{y}$ with the predicted labels $\hat{\mathbf{y}}$. In our work, we take the loss function to be KLD, that is,[6]

$$\Delta_{KLD}(TP, FP, FN, TN) = \boldsymbol{KLD}(\lambda, \hat{\lambda}), \tag{14}$$

where $\lambda(c) = \frac{TP+FN}{TP+FP+FN+TN}$ and $\hat{\lambda}_{Te}(c) = \frac{TP+FP}{TP+FP+FN+TN}$.

(3) An algorithm for the efficient computation of a hypothesis

$$\hat{\Phi}(\mathbf{x}) = \text{argmax}_{\hat{\mathbf{y}} \in \mathcal{Y}} \{\mathbf{w} \cdot \Psi(\mathbf{x}, \hat{\mathbf{y}})\}, \tag{15}$$

where $\mathbf{w}$ is a vector of parameters. In SVM$_{perf}$, this simply corresponds to computing

$$\hat{\Phi}(\mathbf{x}) = (\text{sign}(\mathbf{w} \cdot x_1), \ldots, \text{sign}(\mathbf{w} \cdot x_n)). \tag{16}$$

(4) An algorithm for the efficient computation of the *loss-augmented* hypothesis

$$\hat{\Phi}_{\Delta}(\mathbf{x}) = \text{argmax}_{\hat{\mathbf{y}} \in \mathcal{Y}} \{\Delta(\mathbf{y}, \hat{\mathbf{y}}) + \mathbf{w} \cdot \Psi(\mathbf{x}, \hat{\mathbf{y}})\}, \tag{17}$$

which in SVM$_{perf}$ is computed via an algorithm [Joachims 2005, Algorithm 2] with $O(n^2)$ worst-case complexity.

We have used the implementation of SVM$_{perf}$ made available by Joachims,[7] which we have extended by implementing the module that takes care of the Δ_{KLD} loss function. In the rest of the article, our method will be dubbed SVM(KLD).

4. EXPERIMENTS

We now present the results of experiments aimed at assessing whether the approach we have proposed in Section 3 delivers better quantification accuracy than state-of-the-art quantification methods. In order to do this, we have run all our experiments by using as baselines for our SVM(KLD) method all the methods described in Section 2.2.

[5]For this formulation of Ψ, and when error rate is the chosen loss function, Joachims [2005] shows that SVM$_{perf}$ coincides with the traditional univariate SVM model (called SVM$_{org}$ in Joachims [2005]).

[6]In Equation (14), ***KLD*** is written as a function of TP, FP, FN, TN for the simple fact that in Joachims [2005] (where SVM$_{perf}$ was originally described), the loss function Δ is specified as a function of the four cells of the contingency table. However, it should be clear that $\lambda(c)$ does not depend on the predicted labels: even if in Equation (14) we have written it out as $\lambda(c) = \frac{TP+FN}{TP+FP+FN+TN}$, this latter is equivalent to writing $\lambda(c) = \frac{GP}{GP+GN}$, where GP (the "gold positives") is $TP + FN$ and GN (the "gold negatives") is $FP + TN$. Seen under this light, there is no trace of predicted labels in $\lambda(c) = \frac{GP}{GP+GN}$, and $\lambda(c)$ is just a function of the gold standard and not of the prediction. Analogously, it should be clear that $\hat{\lambda}(c)$ is just a function of the prediction and not of the gold standard.

[7]SVM$_{perf}$ is available from http://www.cs.cornell.edu/People/tj/svm%5Flight/svm_perf.html. Our module that extends it to deal with KLD is available at http://hlt.isti.cnr.it/quantification/.

For the CC, ACC, T50, X, MAX, MS, MM(KS), and MM(PP) methods, we have used the original implementation that we have obtained from the author (this guarantees that the baselines perform at their full potential). We have instead implemented PCC and PACC ourselves. At the heart of the implementation of all the baselines is a standard linear SVM with the parameters set at their default values; where quantities (such as, e.g., fpr_{Te} and tpr_{Te}—see Equation (8)) had to be estimated from the training set, we have used 50-fold cross-validation, as done and recommended by Forman [2008]. In order to guarantee a fair comparison with the baselines, we have used the default values for the parameters also for SVM$_{perf}$, which lies at the basis of our SVM(KLD) method.[8]

In order to generate the vectorial representations for our documents, the classic "bag-of-words" approach has been adopted. In particular, punctuation has been removed, all letters have been converted to lowercase, numbers have been removed, stop words have been removed using the stop list provided in Lewis [1992, pages 117–118], and stemming has been performed by means of the version of Porter's stemmer available from http://snowball.tartarus.org/. All the remaining stemmed words ("terms") that occur at least once in Tr have thus been used as features of our vectorial representations of documents; no feature selection has been performed. Feature weights have been obtained via the "ltc" variant [Salton and Buckley 1988] of the well-known $tfidf$ class of weighting functions, that is,

$$tfidf(t_k, d_i) = tf(t_k, d_i) \cdot \log \frac{|Tr|}{\#_{Tr}(t_k)}, \tag{18}$$

where d_i is a document, $\#_{Tr}(t_k)$ denotes the number of documents in Tr in which feature t_k occurs at least once, and

$$tf(t_k, d_i) = \begin{cases} 1 + \log \#(t_k, d_i) & \text{if } \#(t_k, d_i) > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{19}$$

where $\#(t_k, d_i)$ denotes the number of times t_k occurs in d_i. Weights obtained by Equation (18) are normalized through cosine normalization, that is,

$$w_{ki} = \frac{tfidf(t_k, d_i)}{\sqrt{\sum_{s=1}^{|\mathcal{T}|} tfidf(t_s, d_i)^2}}, \tag{20}$$

where $\mathcal{T}$ denotes the total number of features. Following Forman [2008], we set the ϵ constant for smoothing KLD to the value $\epsilon = \frac{1}{2|Te|}$.

4.1. Datasets

The datasets we use for our experiments have been extracted from two important text classification test collections, Reuters Corpus Volume 1 version 2 (RCV1-v2) and OHSUMED-S.

RCV1-v2 is a standard, publicly available benchmark for text classification consisting of 804,414 news stories produced by Reuters from August 20, 1996, to August 19, 1997.[9] RCV1-v2 ranks as one of the largest corpora currently used in text classification research and, as pointed out by Forman [2006b], suffers from extensive "drift," that is, from substantial variability between the training set and the test set, which makes it

[8]An additional reason we have left the parameters at their default values is that, in a context in which the characteristics of Tr and Te may substantially differ, it is not clear that the parameter values that are found optimal on Tr via k-fold cross-validation will also prove optimal (or at least will perform reasonably) on Te.
[9]http://trec.nist.gov/data/reuters/reuters.html.

a challenging dataset for quantification. In our experiments, we have used the 12,807 news stories of the first week (August 20 to 26, 1996) for training, and the 791,607 news stories of the other 52 weeks for testing.[10] We have further partitioned these latter into 52 test sets each consisting of one week's worth of data.[11] RCV1-v2 is multilabel; that is, a document may belong to several classes at the same time. Of the 103 classes of which its "Topic" hierarchy consists, in our experiments, we have restricted our attention to the 99 classes with at least one positive training example. Consistently with the evaluation presented by Lewis et al. [2004], classes placed at internal nodes in the hierarchically organized classification scheme are also considered in the evaluation; as positive examples of these classes, we use the union of the positive examples of their subordinate nodes, plus their "own" positive examples.

The OHSUMED-S dataset [Esuli and Sebastiani 2013] is a subset of the well-known OHSUMED test collection [Hersh et al. 1994]. OHSUMED-S consists of a set of 15,643 MEDLINE records spanning the years from 1987 to 1991, where each record is classified under one or more of the 97 MeSH index terms that belong to the *Heart Disease* (HD) subtree of the well-known MeSH tree of index terms.[12] Each entry consists of summary information relative to an article published in one of 270 medical journals; the available fields are title, abstract, author, source, publication type, and MeSH index terms. As the training set we have used, consistently with Esuli and Sebastiani [2013], the 2,510 documents belonging to year 1987; nine MeSH index terms out of the 97 in the HD subtree are never assigned to any training document, so the number of classes we actually use is 88. We partition the four remaining years' worth of data into four bins (1988, 1989, 1990, 1991), each containing the documents generated within the corresponding calendar year. The reason we do not use the entire OHSUMED dataset is that roughly 93% of OHSUMED entries have no class assigned from the HD subtree, which means that the classes in the HD subtree have very low prevalence ($\lambda_{Tr} = 0.003$ on average); we thus prefer to use OHSUMED-S, which presents a wider range of prevalence values.

This experimental setting thus generates $52 \times 99 = 5{,}148$ binary quantification test sets for RCV1-v2 (containing an average of 15,223 documents each) and $4 \times 88 = 352$ test sets for OHSUMED-S (containing an average of 3,283 documents each). This large number of test sets will give us an opportunity to study quantification across different dimensions (e.g., across classes characterized by different prevalence, across classes characterized by different amounts of drift, across time).[13] More detailed figures about our datasets are given in Table I. Note that both RCV1-v2 and OHSUMED-S classes are characterized by severe imbalance, as can be noticed by the two "Avg prevalence of the positive class" rows of Table I, where both values are very far away from the value of 0.5, which represents perfect balance. On a side note, it is well known that the "bag-of-words" extraction process outlined a few paragraphs earlier gives rise to very high-dimensional (albeit sparse) vectors; our case is no exception, and the dimensionality of our vectors is 53,204 (RCV1-v2) and 11,286 (OHSUMED-S), respectively.

Note that the experimental protocol we adopt is different from the one adopted by Forman. Forman [2005, 2006a, 2008] proposes a protocol in which, given a training set Tr and a test set Te, several controlled experiments are run by artificially altering

[10]This is the standard "LYRL2004" split between training and test data, originally defined in Lewis et al. [2004].

[11]More precisely, since the period covered by RCV1-v2 consists of 365 days, that is, 52 full weeks +1 day, the 52nd test set consists of 1 day's worth of data only.

[12]https://www.nlm.nih.gov/mesh/.

[13]In order to guarantee perfect reproducibility of our results, we make available at http://hlt.isti.cnr.it/quantification/ the feature vectors of the RCV1-v2 and OHSUMED-S documents as extracted from our preprocessing module and already split respectively into the 53 and 5 sets described earlier.

Table I. Main Characteristics of the Datasets Used in Our Experiments

		RCV1-v2	OHSUMED-S
ALL	Total # of docs	804,414	15,643
	# of classes (i.e., binary tasks)	99	88
	Time unit used for split	week	year
TRAINING	# of docs	12,807	2,510
	# of features	53,204	11,286
	Min # of positive docs per class	2	1
	Max # of positive docs per class	5,581	782
	Avg # of positive docs per class	397	55
	Min prevalence of the positive class	0.0001	0.0004
	Max prevalence of the positive class	0.4375	0.3116
	Avg prevalence of the positive class	0.0315	0.0218
TEST	# of docs	791,607	13,133
	# of test sets per class	52	4
	Avg # of test docs per set	15,212	3,283
	Min # of positive docs per class	0	0
	Max # of positive docs per class	9,775	1,250
	Avg # of positive docs per class	494	69
	Min prevalence of the positive class	0.0000	0.0000
	Max prevalence of the positive class	0.5344	0.3532
	Avg prevalence of the positive class	0.0323	0.0209

class prevalences (i.e., by randomly removing predefined percentages of the positives or of the negatives) either on Tr or on Te. This protocol is meant to test the robustness of the methods with respect to different "distribution drifts" (i.e., differences between $\lambda_{Tr}(c)$ and $\lambda_{Te}(c)$ of different magnitude) and different class prevalence values. We prefer to opt for a different protocol, one in which "natural" training and test sets are used, without artificial alterations. The reason is that artificial alterations may generate class prevalence values and/or distribution drifts that are simply not realistic (e.g., a situation in which $\lambda_{Tr}(c) = .40$ and $\lambda_{Te}(c) = .01$); conversely, focusing on naturally occurring datasets forces us to come to terms with realistic levels of class prevalence and/or distribution drift. We will thus adopt the latter protocol in all the experiments discussed in this article, "compensating" for the absence of artificial alterations by also studying (see Section 4.2.3) the behavior of our methods separately on test sets characterized by different (and natural) levels of distribution drift.

4.2. Testing Quantification Accuracy

We have run our experiments by learning quantifiers for each class c on the respective training set and testing the quantifiers separately on each of the test sets, using KLD as the evaluation measure. We have done this for all the 99 classes $\times$ 52 weeks in RCV1-v2 and for all the 88 classes $\times$ 4 years in OHSUMED-S, and for all the 10 baseline methods discussed in Section 2.2 plus our SVM(KLD) method.

4.2.1. Analyzing the Results Along the Class Dimension. We first discuss the results according to the class dimension, that is, by averaging the results for each RCV1-v2 class across the 52 test weeks and for each OHSUMED-S class across the 4 years.[14] Since this

[14]Wherever in this article we speak of averaging accuracy results across different test sets, what we mean is actually *macroaveraging*, that is, taking the accuracy results on the individual test sets and computing their arithmetic mean. This is sharply different from *microaveraging*, that is, merging the test sets and computing a single accuracy figure on the merged set. Quite obviously, in a quantification setting, microaveraging does

Table II. Accuracy of SVM(KLD) and of 10 Baseline Methods as Measured in Terms of KLD on the 99 Classes of RCV1-v2 (Top) and on the 88 Classes of OHSUMED-S (Bottom) Grouped by Class Prevalence in *Tr* (Columns 2 to 5); Lower Values Are Better; Column 6 Indicates Average Accuracy across All the Classes
The best result in each column is indicated with **boldface** only when there is a statistically significant difference with respect to each of the other tested methods ($p < 0.001$, two-tailed paired t-test on KLD value across the test sets in the group). The methods are ranked in terms of the value indicated in the "All" column.

		VLP	LP	HP	VHP	All
RCV1-v2	SVM(KLD)	2.09E-03	**4.92E-04**	7.19E-04	1.12E-03	**1.32E-03**
	PACC	2.16E-03	1.70E-03	4.24E-04	2.75E-04	1.74E-03
	ACC	2.17E-03	1.98E-03	5.08E-04	6.79E-04	1.87E-03
	MAX	2.16E-03	2.48E-03	6.70E-04	**9.03E-05**	2.03E-03
	CC	2.55E-03	3.39E-03	1.29E-03	1.61E-03	2.71E-03
	X	3.48E-03	8.45E-03	1.32E-03	2.43E-04	4.96E-03
	PCC	1.04E-02	6.49E-03	3.87E-03	1.51E-03	7.86E-03
	MM(PP)	1.76E-02	9.74E-03	2.73E-03	1.33E-03	1.24E-02
	MS	1.98E-02	7.33E-03	3.70E-03	2.38E-03	1.27E-02
	T50	1.35E-02	1.74E-02	7.20E-03	3.17E-03	1.38E-02
	MM(KS)	2.00E-02	1.14E-02	9.56E-04	3.62E-04	1.40E-02
		VLP	LP	HP	VHP	All
OHSUMED-S	SVM(KLD)	**1.21E-03**	**1.02E-03**	5.55E-04	1.05E-03	**1.13E-03**
	PACC	2.86E-03	2.78E-03	2.82E-04	4.76E-04	2.61E-03
	ACC	2.37E-03	5.40E-03	2.82E-04	**2.57E-04**	2.99E-03
	CC	2.38E-03	8.89E-03	2.82E-03	2.20E-03	4.12E-03
	X	1.38E-03	3.94E-03	3.35E-04	5.36E-03	4.44E-03
	MM(PP)	4.90E-03	1.41E-02	9.72E-04	4.94E-03	7.63E-03
	MM(KS)	1.37E-02	2.32E-02	8.42E-04	5.73E-03	1.14E-02
	MS	3.80E-03	1.79E-03	1.45E-03	1.90E-02	1.18E-02
	T50	7.53E-02	5.17E-02	2.71E-03	1.27E-02	2.74E-02
	MAX	5.57E-03	2.33E-02	1.76E-01	3.78E-01	3.67E-02
	PCC	1.20E-01	8.98E-02	4.93E-02	2.27E-02	1.04E-01

would leave no less than 99 RCV1-v2 classes to discuss, we further average the results across all the RCV1-v2 classes characterized by a training class prevalence $\lambda_{Tr}(c)$ that falls into a certain interval (same for OHSUMED-S and its 88 classes). This allows us to separately check the behavior of our quantification methods on groups of classes that are homogeneous by level of imbalance. We have also run statistical significance tests in order to check whether the improvement obtained by the best-performing method over the second-best performer on the group is statistically significant.[15]

The results are reported in Table II, where four levels of imbalance have been singled out: very low prevalence (VLP, which accounts for all the classes c such that $\lambda_{Tr}(c) < 0.01$; there are 48 RCV1-v2 and 51 OHSUMED-S such classes), low prevalence (LP – $0.01 \leq \lambda_{Tr}(c) < 0.05$; 34 RCV1-v2 and 28 OHSUMED-S classes), high prevalence (HP – $0.05 \leq \lambda_{Tr}(c) < 0.10$; 10 RCV1-v2 and four OHSUMED-S classes), and very high prevalence (VHP – $0.10 \leq \lambda_{Tr}(c)$; seven RCV1-v2 and five OHSUMED-S classes).

The first observation that can be made by looking at the RCV1-v2 results in this table is that, when evaluated across all our 5,148 test sets (Column 6), SVM(KLD) outperforms all the other baseline methods in a statistically significant way, scoring a KLD value of 1.32E-03 against the 1.74E-03 value (a -24.2% error reduction) obtained

not make any sense at all, since false positives from one set and false negatives from another set would compensate each other, thus generating misleadingly high accuracy values.

[15]All the statistical significance tests discussed in this article are based on a two-tailed paired t-test and the use of a 0.001 significance level.

by the best-performing baseline (the PACC method). This is largely a result of a much better balance between false positives and false negatives obtained by the base classifiers: while (as already observed in footnote 3) the average FP/FN ratio across the 5,148 test sets is 0.109 for CC, it is 0.684 for SVM(KLD) (and it is 1 for the perfect quantifier). The OHSUMED-S results essentially confirm the insights obtained from the RCV1-v2 results, with SVM(KLD) again the best of the 11 methods and PACC again the second best; the difference between them is now even higher, with an error reduction of -56.8%.

A second observation that the RCV1-v2 results allow us to make is that SVM(KLD) scores well on all the four groups of classes identified; while it is not always the best method (e.g., it is outperformed by other methods in the HP and VHP groups), it consistently performs well on all four groups. In particular, SVM(KLD) seems to excel at classes characterized by drastic imbalance, as witnessed by the VLP group, where SVM(KLD) is the best performer, and by the LP group, where SVM(KLD) is the best performer in a statistically significant way. In fact, this group of classes seems largely responsible for the excellent overall performance (Column 6) displayed by SVM(KLD), since on the LP group the margin between it and the other methods is large (4.92E-04 against 1.98E-03 of the second-best method), and since the VLP and LP groups altogether account for no less than 82 out of the total 99 classes. This latter fact characterizes most naturally occurring datasets, whose class distribution usually exhibits a power law, with very few highly frequent classes and very many highly infrequent classes. The OHSUMED-S results essentially confirm the RCV1-v2 results, with SVM(KLD) again the best performer on the VLP and LP classes and still performing well, although not being the best, on HP and VHP.

The stability of SVM(KLD) is also confirmed by Table III, which reports, for the same groups of test sets identified by Table II, the variance in KLD across the members of the group. For example, on RCV1-v2, Column 3 reports the variance in KLD across all the 34 classes such that $.01 \leq \lambda_{Tr}(c) \leq 0.05$ and across the 52 test weeks, for a total of $34 \times 52 = 1,768$ test sets. What we can observe from this table is that, when averaged across all the $99 \times 52 = 5,148$ test sets (Column 6), the variance of SVM(KLD) is lower than the variance of all other methods in a statistically significant way. The variance of SVM(KLD) is fairly low in all the four subsets of classes, and particularly so in the subsets of the most imbalanced classes (VLP and LP), in which SVM(KLD) is the best performer in a statistically significant way. The OHSUMED-S results essentially confirm the RCV1-v2 results, with SVM(KLD) the best performer on VLP and LP and still behaving well on HP and VHP.

Concerning the baselines, our results seem to disconfirm the ones reported in Forman [2008], according to which the MS method is the best of the lot, and according to which the ACC method "can estimate the class distribution well in many situations, but its performance degrades severely when the training class distribution is highly imbalanced." In our experiments, instead, MS is substantially outperformed by several baseline methods; ACC is instead a strong contender and (contrary to the statement earlier) especially shines on the subsets of the most imbalanced classes. The results of both Tables II and III clearly show that, on both RCV1-v2 and OHSUMED-S, PACC is the best of the baseline methods presented in Section 2.2.

4.2.2. Analyzing the Results Along the Temporal Dimension. We now analyze the results along the temporal dimension. In order to do this for the RCV1-v2 dataset (OHSUMED-S dataset, respectively), for each of the 52 test weeks (4 test years, respectively) we average the 99 (88, respectively) accuracy results corresponding to the individual classes and check the temporal accuracy trend resulting from these averages. This trend is displayed in Figure 1, where the results of SVM(KLD) are

Table III. Variance of SVM(KLD) and of 10 Baseline Methods as Measured in Terms of KLD on the 99 Classes of RCV1-v2 (Top) and on the 88 Classes of OHSUMED-S (Bottom) Grouped by Class Prevalence in *Tr* (Columns 2 to 5); Column 6 indicates variance across all the classes. **Boldface** represents the best value. The methods are ranked in terms of the value indicated in the "All" column.

		VLP	LP	HP	VHP	All
RCV1-v2	SVM(KLD)	**7.52E-06**	**3.44E-06**	8.94E-07	1.56E-06	**5.68E-06**
	PACC	7.58E-06	2.38E-05	1.50E-06	2.26E-07	1.29E-05
	ACC	1.04E-05	7.43E-06	**4.25E-07**	4.26E-07	8.18E-06
	MAX	8.61E-06	2.27E-05	1.06E-06	**1.66E-08**	1.32E-05
	CC	1.79E-05	1.99E-05	1.96E-06	1.66E-06	1.68E-05
	X	2.21E-05	6.57E-04	2.28E-06	1.06E-07	2.64E-04
	PCC	1.75E-04	1.76E-04	3.56E-05	1.59E-04	9.38E-04
	T50	2.65E-04	4.56E-04	2.43E-04	1.19E-05	3.33E-04
	MM(KS)	3.65E-03	7.81E-04	1.46E-06	4.43E-07	2.10E-03
	MM(PP)	4.07E-03	5.69E-04	6.35E-06	2.66E-06	2.21E-03
	MS	9.36E-03	5.80E-05	1.31E-05	6.18E-06	4.61E-03

		VLP	LP	HP	VHP	All
OHSUMED-S	SVM(KLD)	**3.33E-06**	**8.95E-06**	3.74E-07	3.73E-06	**4.73E-06**
	PACC	9.01E-05	4.88E-05	9.73E-08	4.85E-07	7.12E-05
	ACC	1.06E-05	1.19E-04	**8.92E-08**	**1.34E-07**	4.08E-05
	CC	1.06E-05	1.15E-04	2.74E-06	3.41E-06	4.58E-05
	X	4.72E-06	1.45E-04	1.15E-07	8.72E-05	9.85E-05
	MM(PP)	5.79E-05	5.65E-04	1.93E-06	8.38E-05	2.50E-04
	MM(KS)	2.24E-03	1.02E-03	7.38E-07	1.18E-04	5.55E-04
	MS	2.86E-05	6.43E-06	4.23E-06	2.20E-03	1.35E-03
	T50	1.77E-02	2.83E-03	1.78E-05	2.21E-04	2.23E-03
	MAX	4.38E-04	5.22E-03	5.28E-02	2.89E-01	2.60E-02
	PCC	6.78E-05	9.64E-05	3.78E-05	2.36E-04	7.63E-04

plotted together with the results of the three best-performing baseline methods. The plots unequivocally show that SVM(KLD) is the best method across the entire temporal spectrum for both RCV1-v2 and OHSUMED-S.

Note that quantification accuracy remains fairly stable across time; that is, we are not witnessing any substantial decrease in quantification accuracy with time. Intuition might instead suggest that quantification accuracy should decrease with time, due to the combined effects of true concept drift and distribution drift. This may indicate that (at least in the context of broadcast news that RCV1-v2 represents, and in the context of medical scientific articles that OHSUMED-S represents) the chosen timeframe (1 year for RCV1-v2, 4 years for OHSUMED-S) is not sufficient enough a timeframe to observe a significant such drift.

4.2.3. Analyzing the Results Along the Distribution Drift Dimension. The last angle according to which we analyze the results is distribution "drift." That is, we conduct our analysis in terms of how much the prevalence $\lambda_{Te_i}(c)$ in a given test set Te_i "drifts away" from the prevalence $\lambda_{Tr}(c)$ in the training set. Specifically, for all our 5,148 RCV1-v2 test sets (352 OHSUMED-S test sets, respectively) Te_i we compute $KLD(\lambda_{Te_i}, \lambda_{Tr})$, we rank all the test sets according to the KLD value they have obtained, and we subdivide the resulting ranking into four equally sized segments (quartiles) of 5,148/4 = 1,287 RCV1-v2 test sets (352/4 = 88 OHSUMED-S test sets, respectively) each. As a result, each resulting quartile contains test sets that are homogeneous according to the divergence of their distributions from the corresponding distributions in the training set (different test sets pertaining to the same class c may thus end up in different quartiles). This allows us to investigate how well the different methods behave when

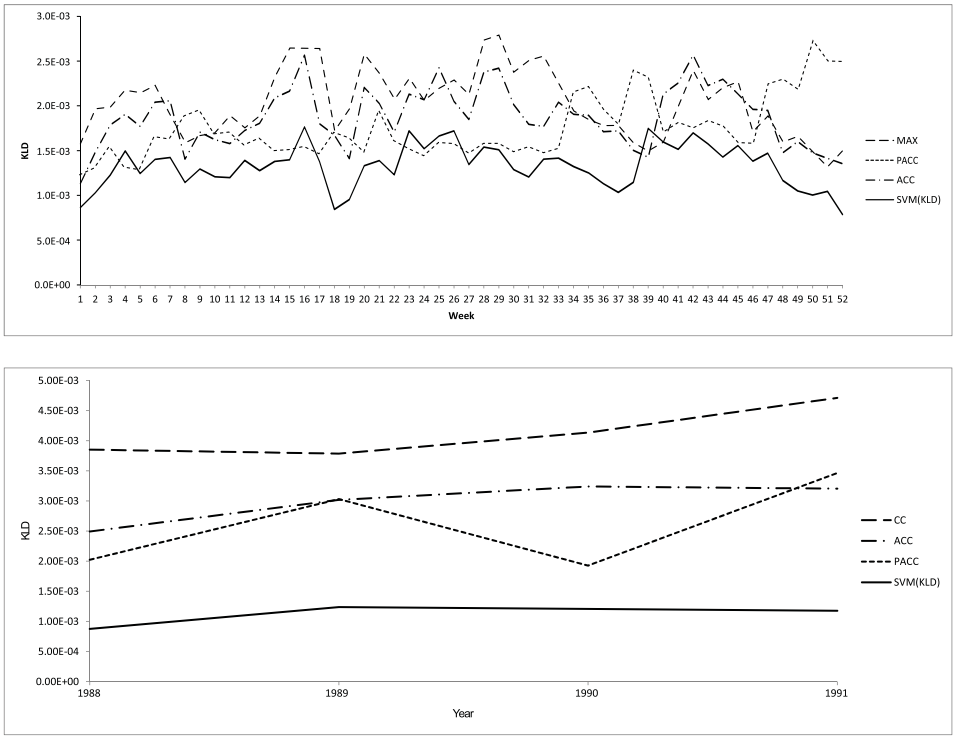

Fig. 1. Values of KLD obtained by SVM(KLD) and by the three best-performing baseline methods on the RCV1-v2 dataset (top) and on the OHSUMED-S dataset (bottom). Each point represents the average KLD value obtained across the 99 RCV1-v2 classes for the given week (top) and across the 88 OHSUMED-S classes for the given year (bottom); lower values are better. Points are plotted as a temporal series (the 52 weeks and the 4 years, respectively, are chronologically ordered).

distribution drift is low (indicated by low KLD values) and when distribution drift is high (high KLD values). We have also run statistical significance tests in order to check whether the improvement obtained by the best-performing method over the second-best performer on the test sets belonging to a certain quartile is statistically significant.

The results of this analysis are displayed in Table IV. The most important observation here is that SVM(KLD) is the best performer in a statistically significant way, both overall and on three out of four quartiles (on the "very high drift" quartile, SVM(KLD) is outperformed by PACC). Additionally, it is worth observing that its performance is consistently high on each quartile. Table V reports variance figures, showing again the stability of SVM(KLD), which is the most stable method overall and on three out of four quartiles (on the VHD quartile the most stable method is PACC).

4.2.4. Evaluating the Results According to RAE. It may be interesting to analyze the results of the previous experiments according to an evaluation measure different from KLD, such as the RAE measure introduced in Section 2.1. As discussed in Section 2.1, RAE is the most (and only) reasonable alternative to KLD proposed so far. Given the simplicity of its mathematical form, it is also a measure everyone can relate to.

Similarly to Table IV, Table VI reports the results of our experiments broken down into quartiles of test sets homogeneous by distribution drift; the difference with Table IV is that RAE is now used as the evaluation measure in place of KLD. The

Table IV. Same as Table II, but with Test Sets Grouped by Distribution Drift Instead of by Class Prevalence

		VLD	LD	HD	VHD	All
RCV1-v2	SVM(KLD)	**1.17E-03**	**1.10E-03**	**1.38E-03**	1.67E-03	**1.32E-03**
	PACC	1.92E-03	2.11E-03	1.74E-03	1.20E-03	1.74E-03
	ACC	1.70E-03	1.74E-03	1.93E-03	2.14E-03	1.87E-03
	MAX	2.20E-03	2.15E-03	2.25E-03	1.52E-03	2.03E-03
	CC	2.43E-03	2.44E-03	2.79E-03	3.18E-03	2.71E-03
	X	3.89E-03	4.18E-03	4.31E-03	7.46E-03	4.96E-03
	PCC	8.92E-03	8.64E-03	7.75E-03	6.24E-03	7.86E-03
	MM(PP)	1.26E-02	1.41E-02	1.32E-02	1.00E-02	1.24E-02
	MS	1.37E-02	1.67E-02	1.20E-02	8.68E-03	1.27E-02
	T50	1.17E-02	1.38E-02	1.49E-02	1.50E-02	1.38E-02
	MM(KS)	1.41E-02	1.58E-02	1.53E-02	1.10E-02	1.40E-02
		VLD	LD	HD	VHD	All
OHSUMED-S	SVM(KLD)	**7.00E-04**	**7.54E-04**	**9.39E-04**	2.11E-03	**1.13E-03**
	PACC	2.91E-03	3.60E-03	3.01E-03	**9.25E-04**	2.61E-03
	ACC	2.54E-03	2.83E-03	2.46E-03	4.14E-03	2.99E-03
	CC	3.31E-03	3.87E-03	3.87E-03	5.43E-03	4.12E-03
	X	5.59E-03	5.74E-03	3.83E-03	2.61E-03	4.44E-03
	MM(PP)	7.82E-03	7.21E-03	7.35E-03	8.14E-03	7.63E-03
	MM(KS)	1.10E-02	1.14E-02	8.66E-03	1.44E-02	1.14E-02
	MS	1.52E-02	9.88E-03	1.50E-02	7.11E-03	1.18E-02
	T50	2.25E-02	3.07E-02	2.30E-02	3.32E-02	2.74E-02
	MAX	5.00E-02	2.88E-02	2.69E-02	4.10E-02	3.67E-02
	PCC	1.10E-01	1.07E-01	9.97E-02	9.88E-02	1.04E-01

Table V. Same as Table III, but with Test Sets Grouped by Distribution Drift Instead of by Class Prevalence

		VLD	LD	HD	VHD	All
RCV1-v2	SVM(KLD)	**2.73E-06**	**2.08E-06**	**3.91E-06**	1.38E-05	**5.68E-06**
	PACC	9.71E-06	2.26E-05	1.15E-05	**7.14E-06**	1.29E-05
	ACC	6.20E-06	6.67E-06	7.97E-06	1.18E-05	8.18E-06
	MAX	1.45E-05	1.32E-05	1.57E-05	9.19E-06	1.32E-05
	CC	1.57E-05	1.57E-05	1.70E-05	1.85E-05	1.68E-05
	X	1.22E-04	1.30E-04	1.28E-04	6.71E-04	2.65E-04
	PCC	7.61E-04	8.38E-04	8.24E-04	8.85E-04	9.38E-04
	T50	2.13E-04	2.42E-04	2.64E-04	6.08E-04	3.33E-04
	MM(KS)	2.12E-03	2.11E-03	2.94E-03	1.25E-03	2.11E-03
	MM(PP)	2.70E-03	3.06E-03	1.93E-03	1.18E-03	2.22E-03
	MS	4.97E-03	9.03E-03	2.38E-03	2.05E-03	4.61E-03
		VLD	LD	HD	VHD	All
OHSUMED-S	SVM(KLD)	**7.59E-07**	**8.31E-07**	**2.47E-06**	1.37E-05	**4.73E-06**
	PACC	1.90E-05	1.59E-04	1.01E-04	**4.16E-06**	7.12E-05
	ACC	1.49E-05	2.27E-05	3.35E-05	9.16E-05	4.08E-05
	CC	1.23E-05	2.88E-05	2.60E-05	1.15E-04	4.58E-05
	X	9.03E-05	1.95E-04	6.46E-05	4.07E-05	9.85E-05
	MM(PP)	2.37E-04	2.50E-04	1.57E-04	3.65E-04	2.50E-04
	MM(KS)	4.24E-04	5.04E-04	1.91E-04	1.10E-03	5.55E-04
	MS	1.57E-03	1.20E-03	1.81E-03	8.01E-04	1.35E-03
	T50	1.05E-03	1.84E-03	1.26E-03	4.76E-03	2.23E-03
	MAX	5.38E-02	7.31E-03	2.06E-02	2.27E-02	2.60E-02
	PCC	5.86E-04	4.92E-04	9.21E-04	9.86E-04	7.63E-04

Table VI. Same as Table II, but with RAE Used as the Evaluation Measure Instead of KLD

		VLD	LD	HD	VHD	All
RCV1-v2	SVM(KLD)	**0.567**	**0.504**	**0.462**	**0.328**	**0.465**
	ACC	0.885	0.761	0.629	0.420	0.674
	CC	1.433	1.174	1.059	0.683	1.087
	PACC	2.235	1.924	1.369	0.402	1.466
	MAX	2.199	1.420	1.213	6.579	2.853
	X	2.339	1.627	1.360	6.726	3.013
	MM(PP)	6.506	6.189	5.258	3.378	5.333
	T50	5.804	5.238	4.667	6.674	5.596
	MM(KS)	7.195	7.375	5.954	4.113	6.160
	PCC	13.989	16.258	5.788	2.017	9.433
	MS	48.139	19.114	9.770	136.755	53.444
		VLD	LD	HD	VHD	All
OHSUMED-S	MM(KS)	1.489	0.770	1.129	**2.922**	**1.577**
	MM(PP)	0.962	0.726	1.115	4.612	1.854
	CC	0.817	0.774	0.601	6.961	2.288
	PACC	1.012	3.326	1.234	8.283	3.464
	ACC	0.831	0.696	0.505	13.726	3.940
	T50	0.897	0.921	0.720	15.903	4.610
	X	1.051	1.035	0.733	19.178	5.499
	SVM(KLD)	**0.601**	**0.543**	**0.413**	22.176	5.933
	MAX	4.249	4.012	1.779	15.331	6.343
	PCC	15.713	10.755	36.040	93.282	38.948
	MS	50.860	13.498	184.874	118.333	91.891

RCV1-v2 results of Table IV confirm the superiority of SVM(KLD) over the baselines, notwithstanding the discrepancy between evaluation measure (RAE) and loss function optimized (KLD). As an average across the 5,148 test sets, SVM(KLD) obtains an average RAE value of 0.465, which improves in a statistically significant way over the 0.674 result obtained by the second-best performer, ACC; the next best-performing methods, CC and PACC, obtain dramatically worse results (1.087 and 1.466, respectively). SVM(KLD) is also the best performer, in a statistically significant way, in each of the four quartiles.

In this case, OHSUMED-S results are substantially different from the RCV1-v2 results, since here the best performer is MM(KS) (a weak contender in all the experiments that we have reported so far), while SVM(KLD) performs much less well. The overall performance of SVM(KLD) is penalized by a bad performance on the VHD quartile, since it is the best performer (and in a statistically significant way) on the other three quartiles. While there is some discrepancy between the outcomes of the RCV1-v2 experiments and those of the OHSUMED-S experiments, we observe that the former might be considered somehow more trustworthy than the latter ones, since RCV1-v2 is much bigger than OHSUMED-S (approximately 60 times more test documents).

Additionally, and more importantly, let us recall that SVM(KLD) is optimized for KLD, and not for RAE. This means that, in keeping with our plan to use classifiers explicitly optimized for the evaluation function used for assessing quantification accuracy, should we really deem RAE to be the "right" such function, we would implement and use SVM(RAE), and not SVM(KLD). In other words, in the RCV1-v2 results of Table VI, SVM(KLD) turns out to be the best performer *despite the fact* that we here use RAE as an evaluation measure.

4.3. Testing Classification Accuracy

As discussed in Section 1, quantification accuracy is related to the classifier's ability to balance false positives and false negatives but is *not* related to its ability to keep their total number low, which is instead a key requirement in standard classification. However, it is fairly natural to expect that a user will trust a quantification method only inasmuch as its good quantification performance results from reasonable classification performance. In other words, a user is unlikely to accept a classifier with good quantification accuracy but bad classification accuracy.

For this reason, we have compared, in terms of *classification* accuracy, SVM(KLD) against the traditional classification-oriented SVMs (i.e., SVM^{org}—see Section 3), with the goal of ascertaining if the former has also a reasonable classification accuracy. Default parameters have been used for both, for the reasons already explained in the first paragraph of Section 4. We have compared the two systems by using the same training set as used in the quantification experiments, and as the test set the union of the 52 test sets used for quantification (i.e., a single test set of 791,607 documents across 99 classes). Evaluation is based on the F_1 measure, both in its microaveraged (F_1^{μ}) and macro-averaged (F_1^{M}) versions.

On the RCV1-v2 dataset, in terms of F_1^{μ}, SVM(KLD) performs slightly worse than SVM^{org} (.755 instead of .777, with a relative decrease of -2.83%), while in terms of F_1^{M}, SVM(KLD) performs slightly better (.440 instead of .433, a relative increase of $+1.61\%$), which indicates that, on highly imbalanced classes, SVM(KLD) is not only a better quantifier but also a better classifier. To witness, on the 48 classes for which $\lambda_{Tr} \leq 0.01$ (i.e., on the most imbalanced classes), we obtain $F_1^{\mu} = .294$ for SVM^{org}, while SVM(KLD) obtains $F_1^{\mu} = .350$ (an increase of $+19.09\%$). The trends on the OHSUMED-S dataset are similar, though with smaller margins. In terms of F_1^{μ}, SVM(KLD) performs slightly worse than SVM^{org} ($F_1^{\mu} = .713$ instead of .722, with a relative decrease of -1.24%), while in terms of F_1^{M}, SVM(KLD) performs slightly better (.405 instead of .398, a relative increase of $+1.76\%$), confirming the insights obtained from RCV1-v2. On the 51 classes for which $\lambda_{Tr} \leq 0.01$ (i.e., on the most imbalanced classes), we obtain $F_1^{\mu} = .443$ for SVM^{org}, while SVM(KLD) obtains $F_1^{\mu} = .456$ (an increase of $+2.93\%$).

All in all, these results show that classifiers trained via SVM(KLD), aside from delivering top-notch quantification accuracy, are also characterized by very good classification accuracy. This makes the quantifiers generated via SVM(KLD) not only accurate but also trustworthy.

4.4. Testing Efficiency

SVM(KLD) also has good properties in terms of sheer efficiency.

Concerning training, Joachims [2005] proves that training a classifier with SVM$_{perf}$ is $O(n^2)$, with n the number of training examples, for any loss function that can be computed from a contingency table (such as KLD indeed is). This is certainly more expensive than training a classifier by means of a standard, linear SVM (which is at the heart of Forman's implementation of all the quantification methods of Section 2.2), since the latter is well known to be $O(sn)$ (with s the average number of nonzero features in the training objects) [Joachims 2006].

However, note that, while SVM(KLD) only requires training a classifier by means of SVM$_{perf}$, setting up a quantifier with any of the methods of Section 2.2 (with the only exception of the simple CC method) requires more than simply training a classifier. For instance, ACC (together with the methods derived from it, such as T50, X, MAX, MS, and PACC) also requires estimating tpr_{Te} and fpr_{Te} on the training set via k-fold cross-validation, which may be expensive; analogously, both MM(KS) and MM(PP)

require estimating D_c^{Te} and $D_{\bar{c}}^{Te}$ via k-fold cross-validation, and the same considerations apply.

In practice, using SVM$_{perf}$ turns out to be affordable. On RCV1-v2, training the 99 binary classifiers described in the previous sections via SVM$_{perf}$ required on average about 4.7 seconds each.[16] By contrast, training the analogous classifiers via a standard linear SVM required on average only 2.1 seconds each. However, this means that, if k-fold cross-validation is used for the estimation of parameters with a value of $k \geq 2$ (meaning that, for each class, additional k classifiers need to be trained), the computational advantage of using a linear SVM instead of the more expensive SVM$_{perf}$ is completely lost. Forman [2008] recommends choosing $k = 50$ in order to obtain more accurate estimates of tpr_{Te} and fpr_{Te} for use in ACC and derived methods; this means making the training phase roughly $(2.1 \cdot 51)/4.7 \approx 22$ times slower than the training phase of SVM(KLD).

Concerning the computational costs involved at classification/quantification time, SVM(KLD) and all the baseline methods discussed in this article are equivalent, since (1) they all generate linear classifiers of equal efficiency and (2) in ACC and derived methods, the cost of the postprocessing involved in computing class prevalences from the classification decisions is negligible.

5. RELATED WORK

An early mention of quantification can be found by Lewis [1995, Section 7], where this task is simply called *counting*; however, the author does not propose any specific solution for this problem. Forman [2005, 2006, 2006a, 2008] is to be credited for bringing the problem of quantification to the attention of the data mining and machine-learning research communities and for proposing several solutions for performing quantification and for evaluating it.

5.1. Applications of Quantification

Chan and Ng [2005, 2006] apply quantification (which they call "class prior estimation") to determine the prevalence of different senses of a word in a text corpus, with the goal of improving the accuracy of word sense disambiguation algorithms as applied on that corpus. Forman [2008] uses quantification in order to establish the prevalence of various support-related issues in incoming telephone calls received at customer support desks. Esuli and Sebastiani [2010a] apply quantification methods for estimating the prevalence of various response classes in open-ended answers obtained in the context of market research surveys (they do not use the term "quantification" and rather speak of "measuring classification accuracy at the aggregate level"). Hopkins and King [2010] classify blog posts with the aim of estimating the prevalence of different political candidates in bloggers' preferences, while Ceron et al. [2014] and Curini et al. [2015] use tweet quantification in order to analyze the online popularity of political leaders and the voting intentions of Internet users. Gonzalez-Castro et al. [2013] and Sanchez et al. [2008] use quantification for establishing the prevalence of damaged sperm cells in a given sample for veterinary applications. Baccianella et al. [2013] classify radiology reports with the aim of estimating the prevalence of different pathologies. Tang et al. [2010] focus on *network* quantification problems, that is, problems in which the goal is to estimate class prevalence among a population of nodes in a network. Alaiz-Rodríguez et al. [2011], Limsetto and Waiyamai [2011], Xue and Weiss [2009], and Zhang and Zhou [2010] use quantification in order to improve classification, that is, attempt to

[16]All times reported in this section were measured on a commodity machine equipped with an Intel Centrino Duo 2×2GHz processor and 2GB RAM.

estimate class prevalence in the test set in order to generate a classifier that better copes with differences in the class distributions of the training set and the test set.

Many other works use quantification "without knowingly doing so"; that is, unaware of the existence of methods specifically optimized for quantification, they use classification with the only goal of estimating class prevalences. In other words, these works use plain "classify and count." Among them, Mandel et al. [2012] use tweet quantification in order to estimate, from a quantitative point of view, the emotional responses of the population (segmented by location and gender) to a natural disaster; O'Connor et al. [2010] analyze the correlation between public opinion as measured via tweet sentiment quantification and via traditional opinion polls; Dodds et al. [2011] use tweet sentiment quantification in order to infer spatiotemporal happiness patterns; and Weiss et al. [2013] use quantification in order to measure the prevalence of different types of pets' activity as detected by wearable devices.

5.2. Quantification Methods

Bella et al. [2010] compare many of the methods discussed in Section 2.2 and find that CC $\prec$ PCC $\prec$ ACC $\prec$ PACC (where $\prec$ means "underperforms"). Also Tang et al. [2010] experimentally compare several among the methods discussed in Section 2.2 and find that CC $\prec$ PCC $\prec$ ACC $\prec$ PACC $\prec$ MS. They also propose a method (specific to linked data) that does not require the classification of individual items, but they find that it underperforms a robust classification-based quantification method such as MS. However, the experimental comparisons of Bella et al. [2010] and Tang et al. [2010] are both framed in terms of absolute error, which seems a substandard evaluation measure for this task (see Section 2.1); additionally, the datasets they test on do not exhibit the severe imbalance typical of many binary text classification tasks, so it is not surprising that their results concerning MS are not confirmed by our experiments.

The idea of using a learner that directly optimizes a loss function specific to quantification was first proposed, although not implemented, by Esuli and Sebastiani [2010b], which indeed proposes using SVM_{perf} to directly optimize KLD; the present article is thus the direct realization of that proposal. The first published work that implements and tests the idea of directly optimizing a quantification-specific loss function is [Milli et al. 2013], who propose variants of decision trees and decision forests that directly optimize a loss-combining classification accuracy and quantification accuracy. At the time of going to print, we have become aware of a related paper [Barranquero et al. 2015] whose authors, following Esuli and Sebastiani [2010b], use SVM_{perf} to perform quantification; differently from the present article, and similarly to Milli et al. [2013], they use an evaluation function that combines classification accuracy and quantification accuracy.

5.3. Other Related Work

Bella et al. [2014] address the problem of performing quantification for the case in which the output variable to be predicted for a given individual is a real value instead of a class as in the case we analyze; that is, they analyze quantification as a counterpart of *regression* rather than of classification, as we do.

Quantification as defined in this article bears some relation with density estimation [Silverman 1986], which can be defined as the task of estimating, based on observed data, the unknown probability density function of a given random variable. If the random variable is discrete, this means estimating, based on observed data, the unknown distribution across the discrete set of events, that is, across the classes. An example density estimation problem is estimating the percentage of white balls in a very large urn containing white balls and black balls. However, there are two essential differences between quantification and density estimation, that is, that (1) in density

estimation, the class to which a data item belongs can be established with certainty (e.g., if a ball is picked, it can be decided with certainty if it is black or white), while in quantification this is not true, and (2) in density estimation, the population of data items is usually so large as to make it infeasible to check the class to which each data item belongs (e.g., only a limited sample of balls is picked), while this is not true in quantification, where it is assumed that all the items can be checked for the purpose of estimating the class distribution.

A research area that might seem related to quantification is *collective classification* (CoC) [Sen et al. 2008]. Similarly to quantification, in CoC, the classification of instances is not viewed in isolation. However, CoC is radically different from quantification in that its focus is on improving the accuracy of classification by exploiting relationships between the objects to classify (e.g., hypertextual documents that link to each other). Differently from quantification, CoC (1) assumes the existence of explicit relationships between the objects to classify, which quantification does not, and (2) is evaluated at the individual level rather than at the aggregate level as quantification.

Quantification bears strong relations with *prevalence estimation from screening tests*, an important task in epidemiology (see Levy and Kass [1970], Lew and Levy [1989], Küchenhoff et al. [2012], Rahme and Joseph [1998], and Zhou et al. [2002]). A screening test is a test that a patient undergoes in order to check if he or she has a given pathology. Tests are often imperfect; that is, they may give rise to false positives (the patient is incorrectly diagnosed with the pathology) and false negatives (the test wrongly diagnoses the patient to be free from the pathology). Therefore, testing a patient is akin to classifying a document, and using these tests for estimating the prevalence of the pathology in a given population is akin to performing aggregative quantification. The main difference between this task and quantification is that a screening test typically has known and fairly constant recall (that epidemiologists call "sensitivity") and fallout (whose complement epidemiologists call "specificity"), while the same usually does not happen for a classifier.

6. CONCLUSIONS

We have presented SVM(KLD), a new method for performing quantification, an important (if scarcely investigated) task in supervised learning, where estimating class prevalence, rather than classifying individual items, is the goal. The method is sharply different from most other methods presented in the literature. While most such methods adopt a general-purpose classifier (where the decision threshold has possibly been tuned according to some heuristics) and adjust the outcome of the "classify and count" phase, we adopt a straightforward "classify and count" approach (with no threshold tuning and/or a posteriori adjustment) but generate a classifier that is directly optimized for the evaluation measure used for estimating quantification accuracy. This is not straightforward, since an evaluation measure for quantification is inherently nonlinear and multivariate, and thus does not lend itself to optimization via standard supervised learning algorithms. We circumvent this problem by adopting a supervised learning method for structured prediction that allows the optimization of nonlinear, multivariate loss functions and extend it to optimize KLD, the standard evaluation measure of the quantification literature.

Experimental results that we have obtained by comparing SVM(KLD) with 10 different state-of-the-art baselines show that SVM(KLD) (1) is more accurate (in a statistically significant way) than the competition; (2) is more stable than the tested baselines, since it systematically shows very good performance irrespectively of class prevalence (i.e., level of imbalance) and distribution drift (i.e., discrepancy between the class distributions in the training and in the test set), (3) is also accurate at the classification (aside from the quantification) level, and (4) is 20 times faster to train than

the competition. These experiments have been run, against 10 state-of-the-art baseline methods, on a batch of 5,148 binary, high-dimensional datasets (averaging more than 15,200 documents each and characterized by more than 50,000 features) and on a further batch of 352 binary, high-dimensional datasets (averaging more than 3,200 documents each and characterized by more than 10,000 features), all characterized by varying levels of class imbalance and distribution drift. These figures qualify the present experimentation as a very large one.

A. APPENDIX: ON THE PRESUMED INVARIANCE OF *TPR* AND *FPR* ACROSS TEST SETS

Most methods described in Section 2.2 (specifically, ACC, PACC, T50, X, MAX, and MS) rely, among other things, on the assumption that tpr_{T_e} and fpr_{T_e} can reliably be estimated via k-fold cross-validation on the training set; in other words, they rely on the assumption that tpr and fpr do not change when the classifier is applied to different test sets.

Indeed, Forman explicitly assumes that the application domains he confronts are of a type that Fawcett and Flach [2005] call "$y \rightarrow \mathbf{x}$ domains," in which tpr and fpr are in fact invariant with respect to the test set the classifier is applied to. The notation $y \rightarrow \mathbf{x}$ means that the value of the y variable (the output label) probabilistically determines the values of the $\mathbf{x}$ variables (the input features); that is, $\mathbf{x}$ causally depends on y. In other words, the class-conditional probabilities $p(\mathbf{x}|y)$ are invariant across different test sets. For instance, our classification problem may consist of predicting whether a patient suffers or not from a given pathology (a fact represented by a binary variable y) given a vector $\mathbf{x}$ of observed symptoms. This is indeed a "$y \rightarrow \mathbf{x}$ domain," since the causality relation is from y to $\mathbf{x}$; that is, it is the presence of the pathology that determines the presence of the symptoms, and not vice versa. In other words, the class-conditional probabilities $p(\mathbf{x}|y)$ do not change across test sets: if more people suffer from the pathology, more people will exhibit its symptoms. Important quantification problems within $y \rightarrow \mathbf{x}$ domains do exist. One of them is when epidemiologists attempt to estimate the prevalence of various causes of death (y) from "verbal autopsies," that is, binary vectors of symptoms ($\mathbf{x}$) as extracted from oral accounts obtained from relatives of the deceased [King and Lu 2008].

However, many application domains are instead of the type that Fawcett and Flach [2005] call "$\mathbf{x} \rightarrow y$ domains," where it is y that causally depends on $\mathbf{x}$, and not vice versa. For instance, our classification problem may consist of predicting whether the home football team is going to win tomorrow's match or not (y) given a number of stats ($\mathbf{x}$) about its recent performance (e.g., number of goals scored in the last k matches, number of goals conceded, etc.). This is indeed an "$\mathbf{x} \rightarrow y$ domain," since the (assumed) causality relation is from $\mathbf{x}$ to y; that is, the recent performance of the team is an indicator of its state of form, which may probabilistically determine the outcome of the game. It is certainly *not* the case that the outcome of the game determines the past performance of the team! And in $\mathbf{x} \rightarrow y$ domains, the class-conditional probabilities $p(\mathbf{x}|y)$ are not guaranteed to be constant.

One may wonder whether *text* classification contexts are $y \rightarrow \mathbf{x}$ or $\mathbf{x} \rightarrow y$ domains. Given the wide array of uses text classification has been put to, we think it is difficult to make general statements about this. We prefer to follow the advice by Fawcett and Flach [2005], who recommend "that researchers test their assumptions in practice." We have thus compared, for each of our 5,148 RCV1-v2 test sets and 352 OHSUMED-S test sets, (1) the tpr_{T_e} and fpr_{T_e} values that the standard linear SVM classifier (the one at the heart of all our baseline methods) has obtained, with (2) the corresponding tpr_{T_r} and fpr_{T_r} values that we have computed on the training set via k-fold cross-validation. If tpr and fpr were invariant across different sets, then (1) and (2) should

Table VII. Average tpr and fpr Values Measured on the Training and Test Sets of the RCV1-v2 and OHSUMED-S Datasets

The values in the avg(tpr_{Tr}) and avg(fpr_{Tr}) are computed via k-fold cross-validation and are thus averages across 99 classes (RCV1-v2) and 88 classes (OHSUMED-S). The values in the avg(tpr_{Te}) and avg(fpr_{Te}) are averages across 5,148 test sets (RCV1-v2) and 352 test sets (OHSUMED-S). Columns 4 and 7 show the relative variation in these values, measured as $(tpr_{Te} - tpr_{Tr})/(tpr_{Tr})$ and $(fpr_{Te} - fpr_{Tr})/(fpr_{Tr})$, respectively.

	avg(tpr_{Tr})	avg(tpr_{Te})	rel % diff	avg(fpr_{Tr})	avg(fpr_{Te})	rel % diff
RCV1-v2	0.443	0.357	−19.29%	2.68E-03	2.54E-03	−5.12%
OHSUMED-S	0.196	0.313	+59.48%	1.41E-03	1.82E-03	+29.09%

be approximately the same. The results, which are displayed in Table VII, show that in our domain, tpr and fpr are far from being invariant across different sets. For instance, on RCV1-v2, the average tpr_{Tr} as computed via k-fold cross-validation is 0.443, while the analogous average on the 5,148 test sets is 0.357, a −19.29% decrease; interestingly enough, on OHSUMED-S, we witness an analogous trend but with opposite sign, with a +59.48% variation (from 0.196 to 0.313) in going from training to test sets.[17] A similar although less marked pattern can be observed for fpr.

In conclusion, tpr and fpr turn out to be far from being invariant across different sets, at least in the application contexts from which our datasets are drawn. It is thus evident that, at the very least in text quantification contexts, assuming that tpr and fpr are indeed invariant, and adopting methods that rely on this assumption, is risky, and definitely suboptimal in some cases. This is yet another reason to prefer methods, such as SVM(KLD), that do not rely on any such assumption.

ACKNOWLEDGMENTS

We are indebted to George Forman for letting us have the code for all the quantification methods he introduced in Forman [2005, 2006a, 2008], which we have used as baselines. We are also very grateful to Thorsten Joachims for making his SVM$_{perf}$ package available and for useful discussions about its use in quantification.

REFERENCES

Rocío Alaíz-Rodríguez, Alicia Guerrero-Curieses, and Jesús Cid-Sueiro. 2011. Class and subclass probability re-estimation to adapt a classifier in the presence of concept drift. *Neurocomputing* 74, 16 (2011), 2614–2623.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2013. Variable-Constraint classification and quantification of radiology reports under the ACR Index. *Expert Systems and Applications* 40, 9 (2013), 3441–3449.

Jose Barranquero, Jorge Díez, and Juan José del Coz. 2015. Quantification-oriented learning based on reliable classifiers. *Pattern Recognition* 48, 2 (2015), 591–604.

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

Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. 2010. Quantification via probability estimators. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM'10)*. 737–742.

Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. 2014. Aggregative quantification for regression. *Data Mining and Knowledge Discovery* 28, 2 (2014), 475–518.

Andrea Ceron, Luigi Curini, Stefano M. Iacus, and Giuseppe Porro. 2014. Every tweet counts? How sentiment analysis of social media can improve our knowledge of citizens' political preferences with an application to Italy and France. *New Media & Society* 16, 2 (2014), 340–358.

Yee Seng Chan and Hwee Tou Ng. 2005. Word sense disambiguation with distribution estimation. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*. 1010–1015.

[17]Note that the average pairwise difference between two sets of values is *greater than or* equal to the difference of their averages; thus, the discrepancy between the training set values and the corresponding test set values may actually be even higher than Table VII shows.

Yee Seng Chan and Hwee Tou Ng. 2006. Estimating class priors in domain adaptation for word sense disambiguation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL'06)*. 89–96.

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. John Wiley & Sons, New York, NY.

Imre Csiszár and Paul C. Shields. 2004. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory* 1, 4 (2004), 417–528.

Luigi Curini, Andrea Ceron, and Stefano M. Iacus. 2015. Using sentiment analysis to monitor electoral campaigns: Method matters. Evidence from the United States and Italy. *Social Science Computer Review* 33, 1 (2015), 3–20.

Peter Sheridan Dodds, Kameron Decker Harris, Isabel M. Kloumann, Catherine A. Bliss, and Christopher M. Danforth. 2011. Temporal patterns of happiness and information in a global social network: Hedonometrics and Twitter. *PLoS ONE* 6, 12 (2011).

Andrea Esuli and Fabrizio Sebastiani. 2010a. Machines that learn how to code open-ended survey data. *International Journal of Market Research* 52, 6 (2010), 775–800.

Andrea Esuli and Fabrizio Sebastiani. 2010b. Sentiment quantification. *IEEE Intelligent Systems* 25, 4 (2010), 72–75.

Andrea Esuli and Fabrizio Sebastiani. 2013. Training data cleaning for text classification. *ACM Transactions on Information Systems* 31, 4 (2013).

Tom Fawcett and Peter Flach. 2005. A response to Webb and Ting's 'On the application of ROC analysis to predict classification performance under varying class distributions'. *Machine Learning* 58, 1 (2005), 33–38.

George Forman. 2005. Counting positives accurately despite inaccurate classification. In *Proceedings of the 16th European Conference on Machine Learning (ECML'05)*. Porto, PT, 564–575.

George Forman. 2006a. Quantifying trends accurately despite classifier error and class imbalance. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (KDD'06)*. Philadelphia, PA, 157–166.

George Forman. 2006b. Tackling concept drift by temporal inductive transfer. In *Proceedings of the 29th ACM International Conference on Research and Development in Information Retrieval (SIGIR'06)*. 252–259.

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

George Forman, Evan Kirshenbaum, and Jaap Suermondt. 2006. Pragmatic text mining: Minimizing human effort to quantify many issues in call logs. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (KDD'06)*. 852–861.

Michael Gamon. 2004. Sentiment classification on customer feedback data: Noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING'04)*. Geneva, CH, 841–847.

Daniela Giorgetti and Fabrizio Sebastiani. 2003. Automating survey coding by multiclass text categorization techniques. *Journal of the American Society for Information Science and Technology* 54, 14 (2003), 1269–1277.

Víctor González-Castro, Rocío Alaiz-Rodríguez, and Enrique Alegre. 2013. Class distribution estimation based on the Hellinger distance. *Information Sciences* 218 (2013), 146–164.

William Hersh, Christopher Buckley, T. J. Leone, and David Hickman. 1994. OHSUMED: An interactive retrieval evaluation and new large text collection for research. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval (SIGIR'94)*. 192–201.

Daniel J. Hopkins and Gary King. 2010. A method of automated nonparametric content analysis for social science. *American Journal of Political Science* 54, 1 (2010), 229–247.

Thorsten Joachims. 2005. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*. 377–384. DOI: http://dx.doi.org/10.1145/1102351.1102399

Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (KDD'06)*. 217–226.

Thorsten Joachims, Thomas Finley, and Chun-Nam Yu. 2009a. Cutting-plane training of structural SVMs. *Machine Learning* 77, 1 (2009), 27–59.

Thorsten Joachims, Thomas Hofmann, Yisong Yue, and Chun-Nam Yu. 2009b. Predicting structured objects with support vector machines. *Communications of the ACM* 52, 11 (2009), 97–104.

Mark G. Kelly, David J. Hand, and Niall M. Adams. 1999. The impact of changing populations on classifier performance. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99)*. 367–371.

Gary King and Ying Lu. 2008. Verbal autopsy methods with multiple causes of death. *Statistical Science* 23, 1 (2008), 78–91.

Helmut Küchenhoff, Thomas Augustin, and Anne Kunz. 2012. Partially identified prevalence estimation under misclassification using the kappa coefficient. *International Journal of Approximate Reasoning* 53, 8 (2012), 1168–1182.

Paul S. Levy and E. H. Kass. 1970. A three-population model for sequential screening for bacteriuria. *American Journal of Epidemiology* 91, 2 (1970), 148–154.

Robert A. Lew and Paul S. Levy. 1989. Estimation of prevalence on the basis of screening tests. *Statistics in Medicine* 8, 10 (1989), 1225–1230.

David D. Lewis. 1992. *Representation and Learning in Information Retrieval*. Ph.D. Dissertation. Department of Computer Science, University of Massachusetts, Amherst, MA. http://www.research.att.com/lewis/papers/lewis91d.ps.

David D. Lewis. 1995. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th ACM International Conference on Research and Development in Information Retrieval (SIGIR'95)*. Seattle, WA, 246–254.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research* 5 (2004), 361–397.

Nachai Limsetto and Kitsana Waiyamai. 2011. Handling concept drift via ensemble and class distribution estimation technique. In *Proceedings of the 7th International Conference on Advanced Data Mining (ADMA'11)*. Bejing, CN, 13–26.

Benjamin Mandel, Aron Culotta, John Boulahanis, Danielle Stark, Bonnie Lewis, and Jeremy Rodrigue. 2012. A demographic analysis of online sentiment during hurricane Irene. In *Proceedings of the NAACL/HLT Workshop on Language in Social Media*. Montreal, CA, 27–36.

Letizia Milli, Anna Monreale, Giulio Rossetti, Fosca Giannotti, Dino Pedreschi, and Fabrizio Sebastiani. 2013. Quantification trees. In *Proceedings of the 13th IEEE International Conference on Data Mining (ICDM'13)*. Dallas, TX, 528–536.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the 4th AAAI Conference on Weblogs and Social Media (ICWSM'10)*.

Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence (Eds.). 2009. *Dataset Shift in Machine Learning*. MIT Press, Cambridge, MA.

Elham Rahme and Lawrence Joseph. 1998. Estimating the prevalence of a rare disease: Adjusted maximum likelihood. *The Statistician* 47 (1998), 149–158.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management* 24, 5 (1988), 513–523.

Claude Sammut and Michael Harries. 2011. Concept drift. In *Encyclopedia of Machine Learning*, Claude Sammut and Geoffrey I. Webb (Eds.). Springer, Berlin, 202–205.

Lidia Sánchez, Víctor González, Enrique Alegre, and Rocío Alaiz. 2008. Classification and quantification based on image analysis for sperm samples with uncertain damaged/intact cell proportions. In *Proceedings of the 5th International Conference on Image Analysis and Recognition (ICIAR'08)*. 827–836.

Robert E. Schapire and Yoram Singer. 2000. BoosTexter: A boosting-based system for text categorization. *Machine Learning* 39, 2/3 (2000), 135–168.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Magazine* 29, 3 (2008), 93–106.

Bernard W. Silverman. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, UK.

Lei Tang, Huiji Gao, and Huan Liu. 2010. Network quantification despite biased labels. In *Proceedings of the 8th Workshop on Mining and Learning with Graphs (MLG'10)*. 147–154.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21 st International Conference on Machine Learning (ICML'04)*.

Gary M. Weiss, Ashwin Nathan, J. B. Kropp, and Jeffrey W. Lockhart. 2013. WagTag: A dog collar accessory for monitoring canine activity levels. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing (UBICOMP'13)*. 405–414.

Jack Chongjie Xue and Gary M. Weiss. 2009. Quantification and semi-supervised classification methods for handling changes in class distribution. In *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD'09)*. 897–906.

ChengXiang Zhai. 2008. Statistical language models for information retrieval: A critical review. *Foundations and Trends in Information Retrieval* 2, 3 (2008), 137–213.

Zhihao Zhang and Jie Zhou. 2010. Transfer estimation of evolving class priors in data stream classification. *Pattern Recognition* 43, 9 (2010), 3151–3161.

Xiao-Hua Zhou, Donna K. McClish, and Nancy A. Obuchowski. 2002. *Statistical Methods in Diagnostic Medicine*. Wiley, New York, NY.

Received April 2013; revised April 2014; accepted October 2014