Optimizing Text Quantifiers for Multivariate Loss Functions

Andrea Esuli and Fabrizio Sebastiani*
Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
[firstname.lastname]@isti.cnr.it

ABSTRACT

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 unlabelled items. Quantification has several applications in IR, 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 generic classifier, counting the unlabelled items which have been assigned the class, and tuning the obtained counts according to some heuristics. In this paper we depart from the tradition of using generic classifiers, and use instead a supervised learning model for *structured prediction*, capable of generating classifiers directly optimized for the (multivariate and non-linear) function used for evaluating quantification accuracy. Experiments on a very large, standard text classification dataset show that this method is more accurate, more stable, and more efficient than existing, state-of-the-art quantification methods.

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 unlabelled data items belong to, but determining the *prevalence* (or "relative frequency") of each class in the unlabelled data. The latter task is known as *quantification* [6, 7, 9, 10].

Although what we are going to discuss here applies to any type of data, we are mostly interested in *text* quantification, i.e., quantification where the data items are textual documents. To see the importance of text quantification, let us examine the task of classifying textual answers returned to open-ended questions in questionnaires [5, 11, 12], 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 a classification scheme which includes 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 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 a classification scheme which includes the class LikedTheCampaign. Here, the agency is likely *not* interested in whether a specific individual belongs to the class LikedTheCampaign, but is likely interested in knowing *how many* respondents belong to it, i.e., 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.

In the absence of methods for estimating class prevalence more directly, the obvious method for dealing with the latter type of scenarios is *quantification via classification*, i.e., classifying each unlabelled document and estimating class prevalence by counting the documents that have been attributed the class. However, there are two reasons why 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

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \quad (1)$$

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 unlabelled data the classifier is supposed to classify. But in real-world settings this assumption is often violated. For instance, in a backlog of newswire stories from year 2001, the prevalence of class Terrorism in August data will likely not be the same

*The order in which the authors are listed is purely alphabetical; each author has given an equally important contribution to this work.

SIGIR 2013, Dublin, IE

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" [23].

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., [1, 6, 7, 9, 10, 14, 31]) employ *generic* supervised learning methods, i.e., address quantification by elaborating on the results returned by a generic classifier. In this paper we take a sharply different, *structured prediction* approach, based upon the use of classifiers explicitly optimized for the non-linear, multivariate evaluation function that we will use for assessing quantification accuracy.

The rest of the paper is organized as follows. In Section 2, after setting the stage we describe the evaluation function we will adopt (§2.1), and we sketch a number of quantification methods previously proposed in the literature (§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 a very large, standard textual dataset, using all the methods of §2.2 as baselines. Section 5 discusses related work, while Section 6 concludes, sketching avenues for further research.

2. PRELIMINARIES

In this paper 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 non-membership in c, respectively. The approaches we will focus on are based on quantification via classification, i.e., they rely on the generation of a classifier $\hat{\Phi} : D \rightarrow \{-1, +1\}$ via supervised learning from a training set Tr. We will indicated with Te the test set on which quantification effectiveness is going to be tested.

We define the *prevalence* (or *relative frequency*) λ_{Te} of class c in a set of documents Te as the fraction of members of Te that belong to c, i.e., as

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

Given a set Te of unlabelled documents and a class c, quantification is defined as the task of estimating $\lambda_{Te}(c)$, i.e., 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 §2.1).

The reasons why we focus on *binary* quantification are two-fold:

- 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 support for either of two candidates in the second round of a two-round ("run-off") election.
- A multi-class multi-label problem (also known as an *n-of-m problem*, i.e., 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. Binary quantification methods can thus also be applied to solving quantification in multi-class multi-label contexts..

We instead leave the discussion of quantification in single-label multi-class (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 quantification accuracy.

The simplest such measure is *bias* (B), defined as $B = \hat{\lambda}_{Te}(c) - \lambda_{Te}(c)$ and used in [6, 7, 29]; 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 in [5], which calls it *percentage discrepancy*, and in [1, 6, 7, 13, 26, 29]), defined as $AE = |\hat{\lambda}_{Te}(c) - \lambda_{Te}(c)|$, 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 $RAE = |\hat{\lambda}_{Te}(c) - \lambda_{Te}(c)|/|\lambda_{Te}(c)|$, 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 [6], who uses *normalized cross-entropy*, better known as *Kullback-Leibler Divergence* (KLD – see e.g., [2]). KLD, defined as

$$KLD(\lambda||\hat{\lambda}) = \sum_{c \in C} \lambda(c) \log \frac{\lambda(c)}{\hat{\lambda}(c)} \quad (3)$$

and also used in [5, 7, 9, 29], is a measure of the error made in estimating a true distribution λ over a set C of classes by means of a distribution $\hat{\lambda}$; 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 and $\hat{\lambda}$) and $+\infty$ (total divergence of λ and $\hat{\lambda}$). In the binary case in which $C = \{c, \overline{c}\}$, KLD becomes

$$KLD(\lambda||\hat{\lambda}) = \lambda_{Te}(c) \log \frac{\lambda_{Te}(c)}{\hat{\lambda}_{Te}(c)} + \lambda_{Te}(\overline{c}) \log \frac{\lambda_{Te}(\overline{c})}{\hat{\lambda}_{Te}(\overline{c})} \quad (4)$$

Note that, as from Equation 3, KLD is undefined when the predicted distribution $\hat{\lambda}$ 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 3 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 and $\hat{\lambda}$ coincide.

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

KLD offers several advantages with respect to RAE (and, *a fortiori*, to B and AE). One advantage is that, as evident from Equation 4, it is symmetric with respect to the complement of a class, i.e., switching the role of c and $\overline{c}$ does not change the result. This means that, e.g., 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 e.g., Employed vs. Unemployed. A second advantage is that KLD is not defined only on the binary (and multi-label multi-class) case, but is also defined on the single-label multi-class 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 [3] and, although from a more applicative angle, within the language modelling approach to information retrieval [32].

2.2 Existing quantification methods

A number of methods have been proposed in the (still brief) literature on quantification; below 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, i.e.,

$$\hat{\lambda}^{CC}_{Te}(c) = \frac{|\{d_j \in Te | \hat{\Phi}(d_j) = +1\}|}{|Te|} \tag{5}$$

Forman et al. [9, 10] call this the *classify and count* (CC) method.

Adjusted Classify and Count (ACC). Forman [6, 9] uses a further method which he calls "Adjusted Count", and which we will call *Adjusted Classify and Count* (ACC) so as to make its relation with CC more explicit. The underlying idea is that CC would be optimal, were it not for the fact that an imperfect classifier generates some false positives and false negatives, and that their presence leads 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:

$$\lambda^{ACC}_{Te}(c) = \frac{\hat{\lambda}^{CC}_{Te}(c) - fpr_{Te}}{tpr_{Te} - fpr_{Te}} \tag{6}$$

Since we cannot know the true values of tpr_{Te} and fpr_{Te}, the ACC method consists of estimating them on Tr via k-fold cross-validation and using the resulting estimates in Equation 6.

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} and fpr_{Te} may be imperfect. This lead Forman [9] 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].

Threshold@0.50 (T50), Method X (X), and Method Max (MAX). Forman [9] 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, [9] recommends to heuristically set the decision threshold in such a way that tpr_{Tr} (as obtained via k-fold cross-validation) is equal to .50 before computing Equation 6. This method is dubbed *Threshold@0.50* (T50). Alternative heuristics that [9] 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, [9] recommends to compute $\lambda^{ACC}_{Te}(c)$ 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 $\lambda^{ACC}_{Te}(c)$. This method is dubbed *Median Sweep* (MS).

Mixture Model (MM). The MM method (proposed in [6]) consists of assuming that the distribution D^{Te} of the scores that the classifier assigns to the test examples is a mixture

$$D^{Te} = \lambda_{Te}(c) \cdot D^{Te}_{c} + (1 - \lambda_{Te}(c)) \cdot D^{Te}_{\overline{c}} \tag{7}$$

where D^{Te}_{c} and $D^{Te}_{\overline{c}}$ 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 $\lambda_{Te}(\overline{c})$ are the parameters of this mixture. The MM method consists of estimating D^{Te}_{c} and $D^{Te}_{\overline{c}}$ 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 [6], which differ in terms of how the goodness of fit between the left- and the right-hand side of Equation 7 is estimated. See [6] for more details.

3. OPTIMIZING FOR QUANTIFICATION ACCURACY

A problem with the methods discussed in §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 is seems to implicitly rely on the hypothesis that estimating D^{Te}_{c} and $D^{Te}_{\overline{c}}$ 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 training set and test set may have in the application contexts where quantification is required? 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 paper we propose a new, theoretically well-founded quantification method that radically differs from the ones discussed in §2.2. Note that all of the methods discussed in §2.2 employ *generic* supervised learning methods, i.e., address quantification by post-processing 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, 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 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[2].

We take a sharply different approach, based upon 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 non-trivial, because the evaluation measure that we want to use (KLD) is non-linear, i.e., 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* non-linear, 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 non-linear, and should thus be multivariate, i.e., take in consideration all test items at once.

As discussed in [15], 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 [15][3]. SVM^{perf} is a learning algorithm of the Support Vector Machine family that can generate classifiers optimized for any non-linear, 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 [17, 18, 30] for "structured prediction", i.e., 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 in [18], 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, 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 to generalise not only on inputs ($\mathbf{x}$) but also on outputs ($\mathbf{y}$), thus allowing to produce predictions not seen in the training data.

   In SVM^{perf} Ψ is defined[4] as

   $$\Psi(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} y_i \mathbf{x_i} \quad (8)$$

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, i.e.,

   $$\Delta_{KLD}(TP, FP, FN, TN) = KLD(\lambda(c)||\hat{\lambda}(c)) \quad (9)$$

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

3. An algorithm for the efficient computation of a hypothesis

   $$\hat{\Phi}(\mathbf{x}) = \text{argmax}_{\hat{\mathbf{y}} \in \mathcal{Y}} \{\mathbf{w} \cdot \Psi(\mathbf{x}, \hat{\mathbf{y}})\} \quad (10)$$

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

   $$\hat{\Phi}(\mathbf{x}) = (\text{sign}(\mathbf{w} \cdot x_1), \ldots, \text{sign}(\mathbf{w} \cdot x_n)) \quad (11)$$

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

   $$\hat{\Phi}_{\Delta}(\mathbf{x}) = \text{argmax}_{\hat{\mathbf{y}} \in \mathcal{Y}} \{\Delta(\mathbf{y}, \hat{\mathbf{y}}) + \mathbf{w} \cdot \Psi(\mathbf{x}, \hat{\mathbf{y}})\} \quad (12)$$

   which in SVM^{perf} is computed via an algorithm ([15, Algorithm 2]) with $O(n^2)$ worst-case complexity.

[2]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.

[3]In [15] 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.

[4]For this formulation of Ψ, and when error rate is the chosen loss function, Joachims [15] shows that SVM^{perf} coincides with the traditional univariate SVM model (called SVM_{org} in [15]).

We have used the implementation of SVM^{perf} made available by Joachims[5], which we have extended by implementing the module that takes care of the Δ_{KLD} loss function. In the rest of the paper our method will be dubbed SVM(KLD).

4. EXPERIMENTS

We have run our experiments by using as baselines for our SVM(KLD) method all the methods described in §2.2, in their original implementation that we have obtained from the author (this guarantees that the baselines perform at their full potential). At the heart of their implementation 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 6) had to be estimated from the training set, we have used 50-fold cross-validation, as done in [9]. 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[6]. As a vectorial representation for the documents we have used standard bag-of-words with cosine-normalized $tfidf$ weighting, stop word removal, and no stemming. Following [9], we set the ϵ constant for smoothing KLD to the value $\epsilon = \frac{1}{2}|Te|$.

4.1 Dataset(s)

The dataset we use for our experiments is Reuters Corpus Volume 1 version 2 (RCV1-v2)[7], a standard multi-class multi-label text classification benchmark made available by Reuters and consisting of 804,414 news stories produced by Reuters from 20 Aug 1996 to 19 Aug 1997. RCV1-v2 ranks as one of the largest standard datasets currently used in text classification research. In our experiments we have used the 12,807 news stories of the 1st week (20 to 26 Aug 1996) for training, and the 791,607 news stories of the other 52 weeks for testing (this is the standard "LYRL2004" split between training and test data, originally defined in [22]), split into 52 test sets each consisting of one week's worth of data[8]. As pointed out in [8], RCV1-v2 suffers from extensive "drift", i.e., from substantial variability between the training set and the test set, and it is thus a challenging dataset for quantification. Of the 103 "Topic" classes, in our experiments we have restricted our attention to the 99 classes with at least one positive training example. Consistently with the evaluation presented in [22], also classes placed at internal nodes in the hierarchically organized classification scheme are 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. This setting thus generates $99 \times 52 = 5,148$ binary quantification test sets (containing an average of 15,223 documents each), which will give us an opportunity to study quantification across different dimensions (e.g., across classes, across temporal intervals)[9].

Table 1: Main characteristics of the dataset used in our experiments.

		RCV1-v2
All	Total # of docs	804,414
	# of classes (i.e., binary tasks)	99
	Time unit used for split	week
Training	# of docs	12,807
	Min # of positive docs per class	2
	Max # of positive docs per class	5,581
	Avg # of positive docs per class	397
	Min prevalence of the positive class	0.0001
	Max prevalence of the positive class	0.4375
	Avg prevalence of the positive class	0.0315
Test	# of docs	791,607
	# of test sets per class	52
	Avg # of test docs per set	15,212
	Min # of positive docs per class	0
	Max # of positive docs per class	9775
	Avg # of positive docs per class	494
	Min prevalence of the positive class	0.0000
	Max prevalence of the positive class	0.5344
	Avg prevalence of the positive class	0.0323

More detailed data about our dataset are given in Table 1. Note that these data are, on average, characterized by severe class imbalance, as can be noticed by the two "Avg prevalence of the positive class" rows of Table 1, where both values are very far away from the value of 0.5, which represents perfect balance.

Note that the experimental protocol we adopt is different from the one adopted by Forman. In [6, 7, 9] he proposes a protocol in which, given a training set Tr and a test set Te, several controlled experiments are run by artificially altering class prevalences (i.e., by randomly removing a predefined percentage 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). 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 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 distribution drift. We will thus adopt the latter protocol in all the experiments discussed in this paper, "compensating" for the absence of artificial alterations by also studying (see §4.2.3) the behaviour of our methods separately on test sets characterized by different (and natural) levels of distribution

[5] 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 [URL removed to preserve anonymity].

[6] An additional reason why 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 which are found optimal on Tr via k-fold cross-validation will also prove optimal (or at least will perform reasonably) on Te.

[7] `http://trec.nist.gov/data/reuters/reuters.html`

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

[9] In order to guarantee perfect reproducibility of our results, we make available at [URL removed to preserve anonymity] the feature vectors of the RCV1-v2 documents as extracted from our preprocessing module, and already split into the 53 sets described above.

drift.

4.2 Testing quantification accuracy

We have run our experiments by learning a quantifier for class c on the training set and testing it separately on each of the 52 test sets, using KLD as the evaluation measure. We have done this for all 99 classes in RCV1-v2, for all the baseline methods discussed in §2.2, and for our SVM(KLD) method.

4.2.1 Analyzing the results along the class dimension

We first discuss the results according to the class dimension, i.e., by averaging the results for each class across the 52 test weeks[10]. Since this would leave no less than 99 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. This allows us to separately check the behaviour 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 2nd best performer on the group is statistically significant[11]. The results are reported in Table 2, where four levels of imbalance have been singled out: very low prevalence (VLP – $\lambda_{Tr}(c) < 0.01$, 48 classes), low prevalence (LP – $0.01 \leq \lambda_{Tr}(c) < 0.05$, 34 classes), high prevalence (HP – $0.05 \leq \lambda_{Tr}(c) < 0.10$, 10 classes), and very high prevalence (VHP – $0.10 \leq \lambda_{Tr}(c)$, 7 classes).

The first observation that can be made by looking at 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 1.87E-03 value (a 29.9% error reduction) obtained by the best-performing baseline (the ACC 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 2) the average FP/FN ratio across the 5,148 test sets is 0.109 for CC, it is 0.684 for SVM(KLD).

A second observation 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 2nd best method), and since the VLP and LP groups altogether account for no less than 82 out of the total 99 classes.

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

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

Table 2: Accuracy of SVM(KLD) and of other baseline methods as measured in terms of KLD on the 99 classes of RCV1-v2 grouped by class prevalence in Tr (Columns 2 to 5); lower values are better; Column 6 indicates average accuracy across all the 99 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).

	VHP	HP	LP	VLP	All
SVM(KLD)	2.09E-03	**4.92E-04**	7.19E-04	1.12E-03	**1.32E-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
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

The stability of SVM(KLD) is also confirmed by Table 3, which reports, for the same groups of test sets identified by Table 2, the variance in KLD across the members of the group. For example, Column 3 reports the variance in KLD across all the 34 RCV1-v2 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.

Concerning the baselines, our results seem to disconfirm the ones reported in [9] 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 the best of the tested baselines, and (contrary to the statement above) especially shines on the subsets of the most imbalanced classes.

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 each of the 52 test weeks we average the 99 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 plotted together with the results of the three best-performing baseline methods (the plots of all the other baselines fall off the figure and are thus not displayed). The plots unequivocally show that

Table 3: Variance of SVM(KLD) and of the other baseline methods as measured in terms of KLD on the 99 classes of RCV1-v2 grouped by class prevalence in Tr (Columns 2 to 5); Column 6 indicates variance across all the 99 classes. Boldface represents the best value.

	VHP	HP	LP	VLP	All
SVM(KLD)	**7.52E-06**	**3.44E-06**	8.94E-07	1.56E-06	**5.68E-06**
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
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

SVM(KLD) is the best method across the entire temporal spectrum.

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 $5,148$ test sets 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 $5,148/4 = 1,287$ test sets 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 be part of different quartiles). This allows us to investigate how well the different methods behave when 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 2nd best performer on the test sets belonging to a certain quartile is statistically significant.

The results of this analysis are displayed in Table 4. 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 slightly outperformed by MAX). Additionally, it is worth observing that its performance is consistently high on each quartile. Table 5 reports variance figures, showing again the stability of SVM(KLD).

A second observation is that, while SVM(KLD) is relatively stable, most other methods show a markedly decreasing performance when distribution drift decreases. In other words, most baseline methods are at ease when the distribution drift is substantial but much less so when it is small, i.e., when the distribution in the test set very much resembles that in the training set. This is obviously a plus for SVM(KLD) since, when we need to apply a quantification method to an unlabelled set, we have no *a priori* knowledge of how substantial the distribution drift is, and if there is any at all; a method the quantifies well both in the presence and in the absence of drift is thus highly desirable.

Table 4: Accuracy of SVM(KLD) and of other baseline methods as tested on quartiles of test sets homogeneous by distribution drift. The quartiles are indicated as VHD (very high drift), HD (high drift), LD (low drift), VLD (very low drift). Values indicated are the average of KLD across the test sets belonging to the quartile. The meaning of boldface is the same as for Table 3.

	VHD	HD	LD	VLD	All
SVM(KLD)	1.66E-03	**1.38E-03**	**1.09E-03**	**1.1E-03**	**1.32E-03**
ACC	2.13E-03	1.92E-03	1.74E-03	1.69E-03	1.87E-03
MAX	1.51E-03	2.25E-03	2.14E-03	2.19E-03	2.03E-03
CC	3.17E-03	2.79E-03	2.44E-03	2.43E-03	2.71E-03
X	7.45E-03	4.30E-03	4.18E-03	3.89E-03	4.96E-03
MM(PP)	1.00E-02	1.32E-02	1.40E-02	1.26E-02	1.24E-02
MS	8.67E-03	1.20E-02	1.66E-02	1.37E-02	1.27E-02
T50	1.49E-02	1.48E-02	1.38E-02	1.17E-02	1.38E-02
MM(KS)	1.10E-02	1.52E-02	1.58E-02	1.41E-02	1.40E-02

Table 5: Variance of SVM(KLD) and of other baseline methods as measured in terms of KLD on the $5,148$ test sets grouped by distribution drift (Columns 2 to 5); Column 6 indicates variance across all the $5,148$ test sets. Boldface represents the best value.

	VHD	HD	LD	VLD	All
SVM(KLD)	1.38E-05	**3.91E-06**	**2.07E-06**	**2.72E-06**	**5.68E-06**
ACC	1.17E-05	7.97E-06	6.66E-06	6.19E-06	8.18E-06
MAX	**9.18E-06**	1.57E-05	1.32E-05	1.44E-05	1.32E-05
CC	1.85E-05	1.70E-05	1.56E-05	1.57E-05	1.68E-05
X	6.70E-04	1.28E-04	1.29E-04	1.22E-04	2.64E-04
T50	6.07E-04	2.64E-04	2.42E-04	2.12E-04	3.33E-04
MM(KS)	1.24E-03	2.94E-03	2.11E-03	2.12E-03	2.10E-03
MM(PP)	1.17E-03	1.92E-03	3.06E-03	2.70E-03	2.21E-03
MS	2.05E-03	2.38E-03	9.02E-03	4.96E-03	4.61E-03

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 §2.1. As discussed in §2.1, RAE is the most reasonable alternative to KLD. Given the simplicity of its mathematical form, it is also a measure everyone can relate to.

Similarly to Table 4, Table 6 reports the results of our experiments broken down into quartiles of test sets homogeneous by distribution drift; the difference with Table 4 is that RAE is now used as the evaluation measure in place of KLD. The results of Table 4 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, MAX and CC, obtain dramatically worse results (1.087 and 2.853, respectively). SVM(KLD) is also the best performer, in a statistically significant way, in each of the four quartiles.

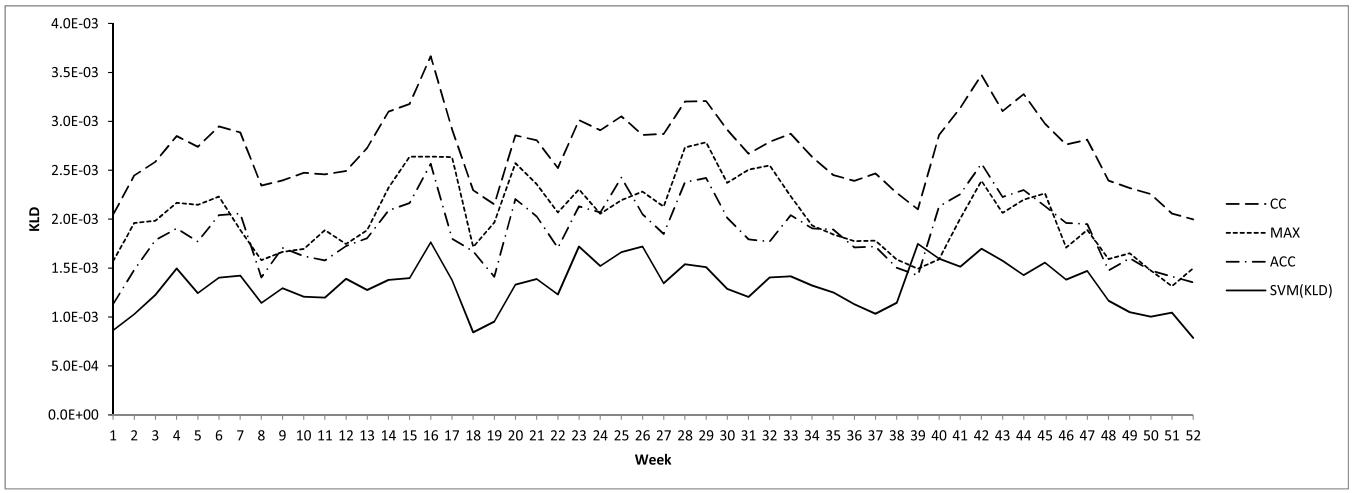

Figure 1: Values of KLD obtained by SVM(KLD) and by the three best-performing baseline methods. Each point represents the average KLD value obtained across the 99 classes for the given week; lower values are better. Points are plotted as a temporal series (the 52 weeks are chronologically ordered).

Table 6: Accuracy of SVM(KLD) and of other baseline methods as tested on quartiles of test sets homogeneous by distribution drift. The quartiles are indicated as VHD (very high drift), HD (high drift), LD (low drift), VLD (very low drift). Values indicated are the average of RAE across the test sets belonging to the quartile. The meaning of boldface is the same as for Table 3.

	VHD	HD	LD	VLD	All
SVM(KLD)	**0.328**	**0.462**	**0.504**	**0.567**	**0.465**
ACC	0.420	0.629	0.761	0.885	0.674
CC	0.683	1.059	1.174	1.433	1.087
MAX	6.579	1.213	1.420	2.199	2.853
X	6.726	1.360	1.627	2.339	3.013
MM(PP)	3.378	5.258	6.189	6.506	5.333
T50	6.674	4.667	5.238	5.804	5.596
MM(KS)	4.113	5.954	7.375	7.195	6.160
MS	136.755	9.770	19.114	48.139	53.444

4.3 Testing efficiency

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

Concerning training, Joachims [15] 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 §2.2), since this latter is well-known to be $O(sn)$ (with s the average number of non-zero features in the training objects) [16].

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 §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, and MAX) 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. Training the 99 binary classifiers described in the previous sections via SVM_{perf} required on average about 4.7 seconds each[12]. By contrast, training the analogous classifiers via a standard linear SVM required on average 2.1 seconds each. This means that, if k-fold cross-validation is used 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[13].

Concerning the computational costs involved at classification / quantification time, SVM(KLD) and all the baseline methods discussed in this paper are equivalent, since they all generate linear classifiers of equal efficiency.

5. RELATED WORK

Forman [6, 7, 9, 10] 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. An early mention of this problem, however, can be found in [21, Section 7], where this task is simply called *counting*.

5.1 Applications of quantification

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

[13]In [9] Forman recommends choosing $k = 50$ in order to obtain more accurate estimates of tpr_{Te} and fpr_{Te}; this means making the training phase roughly $(2.1 \cdot 51)/4.7 \approx 22$ times slower than the training phase of SVM(KLD).

Forman [9] 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 [4] 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 [14] classify blog posts with the aim of estimating the prevalence of different political candidates in bloggers' preferences, while Gonzalez-Castro et al. [13] and Sanchez et al. [26] deal with establishing the prevalence of damaged sperm cells in a given sample for veterinary applications. Tang et al. [29] focus on *network* quantification problems, i.e., problems in which the goal is to estimate class prevalence among a population of nodes in a network. Xue et al. [31] use quantification in order to improve classification, i.e., attempt to 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.

5.2 Quantification methods

Bella et al. [1] compare many of the methods discussed in §2.2, and find that CC < PCC < ACC < PACC (where < means "underperforms"), where PCC and PACC are probabilistic versions of the CC and ACC methods, respectively. Also Tang et al. [29] experimentally compare many of the methods discussed in §2.2, and find that CC < PCC < ACC < PACC < MS. They also propose a method (specific to linked data) that does not require classification, but they find that it underperforms a robust classification-based quantification method such as MS. However, the experimental comparisons of [1] and [29] are both framed in terms of absolute error (see §2.1), which seems a sub-standard evaluation measure for this task; additionally, the datasets they test on do not exhibit the severe imbalance typical of binary text classification, so it is not clear whether their results would carry over to these latter contexts.

5.3 Other related work

Quantification as defined here bears some relation with density estimation [28], which can be defined as the estimation, based on observed data, of 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, i.e., across the classes. An example density estimation problem is to estimate 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, i.e., that (a) 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 (b) in density estimation the population of data items is 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 are checked for the purpose of estimating the class distribution.

A research area that might seem related to quantification is *collective classification* (CoC) [27]. 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 assumes the existence of explicit relationships between the objects to classify (which quantification does not), and 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 [19, 20, 24, 33]). A screening test is a test that a patient undergoes in order to check if s/he has a given pathology. Tests are often imperfect, i.e., 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 quantification via classification. 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 that 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 generic 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 non-linear 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 non-linear, multivariate loss functions, and extend it to optimize KLD, the standard measure of the quantification literature.

Experimental results that we have obtained by comparing SVM(KLD) with state-of-the-art baselines on a very large, standard textual dataset show that SVM(KLD) (i) outperforms the competition, (ii) 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), and (iii) is 20 times faster to train than the competition.

Future work in this area should certainly target quantification in the single-label multi-class case, which the current work does not address.

7. REFERENCES

[1] A. Bella, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Quantification via probability estimators. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM 2010)*, pages 737–742, Sydney, AU, 2010.

[2] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, New York, US, 1991.

[3] I. Csiszár and P. C. Shields. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1(4):417—528, 2004.

[4] A. Esuli and F. Sebastiani. Machines that learn how to code open-ended survey data. *International Journal of Market Research*, 52(6):775–800, 2010.

[5] A. Esuli and F. Sebastiani. Sentiment quantification. *IEEE Intelligent Systems*, 25(4):72–75, 2010.

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

[7] G. Forman. Quantifying trends accurately despite classifier error and class imbalance. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, pages 157–166, Philadelphia, US, 2006.

[8] G. Forman. Tackling concept drift by temporal inductive transfer. In *Proceedings of the 29th ACM International Conference on Research and Development in Information Retrieval (SIGIR 2006)*, pages 252–259, Seattle, US, 2006.

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

[10] G. Forman, E. Kirshenbaum, and J. Suermondt. 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 2006)*, pages 852–861, Philadelphia, US, 2006.

[11] M. Gamon. 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 2004)*, pages 841–847, Geneva, CH, 2004.

[12] D. Giorgetti and F. Sebastiani. Automating survey coding by multiclass text categorization techniques. *Journal of the American Society for Information Science and Technology*, 54(14):1269–1277, 2003.

[13] V. González-Castro, R. Alaiz-Rodríguez, L. Fernández-Robles, R. Guzmán-Martínez, and E. Alegre. Estimating class proportions in boar semen analysis using the Hellinger distance. In *Proceedings of the 23rd International Conference on Industrial Engineering and other Applications of Applied Intelligent Systems (IEA/AIE 2010)*, pages 284–293, Cordoba, ES, 2010.

[14] D. Hopkins and G. King. A method of automated. nonparametric content analysis for social science. *American Journal of Political Science*, 54(1):229—247, 2010.

[15] T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, pages 377–384, Bonn, DE, 2005.

[16] T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, pages 217–226, Philadelphia, US, 2006.

[17] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

[18] T. Joachims, T. Hofmann, Y. Yue, and C.-N. Yu. Predicting structured objects with support vector machines. *Communications of the ACM*, 52(11):97–104, 2009.

[19] P. S. Levy and E. H. Kass. A three-population model for sequential screening for bacteriuria. *American Journal of Epidemiology*, 91(2):148—154, 1970.

[20] R. A. Lew and P. S. Levy. Estimation of prevalence on the basis of screening tests. *Statistics in Medicine*, 8(10):1225–1230, 1989.

[21] D. D. Lewis. Evaluating and optmizing autonomous text classification systems. In *Proceedings of the 18th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1995)*, pages 246–254, Seattle, US, 1995.

[22] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[23] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, editors. *Dataset shift in machine learning*. The MIT Press, Cambridge, US, 2009.

[24] E. Rahme and L. Joseph. Estimating the prevalence of a rare disease: Adjusted maximum likelihood. *The Statistician*, 47:149–158, 1998.

[25] C. Sammut and M. Harries. Concept drift. In C. Sammut and G. I. Webb, editors, *Encyclopedia of Machine Learning*, pages 202–205. Springer, Heidelberg, DE, 2011.

[26] L. Sánchez, V. González, E. Alegre, and R. Alaiz. 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 2008)*, pages 827–836, Póvoa de Varzim, PT, 2008.

[27] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[28] B. W. Silverman. *Density estimation for statistics and data analysis*. Chapman and Hall, London, UK, 1986.

[29] L. Tang, H. Gao, and H. Liu. Network quantification despite biased labels. In *Proceedings of the 8th Workshop on Mining and Learning with Graphs (MLG 2010)*, pages 147–154, Washington, US, 2010.

[30] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21 st International Conference on Machine Learning (ICML 2004)*, Banff, CA, 2004.

[31] J. C. Xue and G. M. Weiss. 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 2009)*, pages 897–906, Paris, FR, 2009.

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

[33] X.-H. Zhou, D. K. McClish, and N. A. Obuchowski. *Statistical Methods in Diagnostic Medicine*. Wiley, New York, US, 2002.