**Social Network Analysis and Mining manuscript No.**
(will be inserted by the editor)

# From Classification to Quantification in Tweet Sentiment Analysis

**Wei Gao · Fabrizio Sebastiani**

Received: March 20, 2016

**Abstract** Sentiment classification has become a ubiquitous enabling technology in the Twittersphere, since classifying tweets according to the sentiment they convey towards a given entity (be it a product, a person, a political party, or a policy) has many applications in political science, social science, market research, and many others. In this paper we contend that most previous studies dealing with tweet sentiment classification (TSC) use a suboptimal approach. The reason is that the final goal of most such studies is not estimating the class label (e.g., Positive, Negative, or Neutral) of individual tweets, but estimating the relative frequency (a.k.a. "prevalence") of the different classes in the dataset. The latter task is called *quantification*, and recent research has convincingly shown that it should be tackled as a task of its own, using learning algorithms and evaluation measures different from those used for classification. In this paper we show (by carrying out experiments using two learners, seven quantification-specific algorithms, and eleven TSC datasets) that using quantification-specific algorithms produces substantially better class frequency estimates than a state-of-the-art classification-oriented algorithm routinely used in TSC. We thus argue that researchers interested in tweet sentiment prevalence should switch to quantification-specific (instead of classification-specific) learning algorithms and evaluation measures.

This is an extended version of a paper with the title "Tweet Sentiment: From Classification to Quantification" which appears in the Proceedings of the 6th ACM/IEEE International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2015).

Both authors are at Qatar Computing Research Institute, Hamad bin Khalifa University, Doha, Qatar; E-mail: {wgao,fsebastiani}@qf.org.qa. Fabrizio Sebastiani is on leave from Consiglio Nazionale delle Ricerche, Italy.

## 1 Introduction

Sentiment classification is the task of detecting, given an opinion-laden textual item (e.g., a product review, a blog post, an editorial, etc.), whether it expresses a positive or a negative opinion about a given entity (e.g., a product, a person, a political party, or a policy). The above scenario is a simple instance of *binary* classification, with Positive and Negative as the classes. Slightly more complex scenarios result when the Neutral class is added to the picture, which makes the task an instance of *single-label multi-class* (SLMC) classification, or when sentiment strength needs to be assessed on an ordered scale consisting of VeryPositive, Positive, Fair, Negative, VeryNegative, which makes the task one of *ordinal classification*.

In any of the above incarnations, sentiment classification has become a ubiquitous enabling technology in the Twittersphere, since classifying tweets according to the sentiment they convey towards a given entity has many applications in political science, social science, market research, and many others [41,42]. The tweet sentiment classification (TSC) shared task which has taken place in the context of the last three SemEval evaluation campaigns (where it is called "Sentiment Analysis in Twitter" – see [47,53,54]) has been, in all three editions, the SemEval task with the highest number of participants.

In this paper we contend that most previous studies dealing with TSC use a suboptimal approach. The rest of this section is devoted to arguing why this is so.

Usually, the final goal of most such studies is *not* estimating the label of an individual tweet, but studying the distribution of a set of tweets across the classes of interest; in other words, the interest in such studies is not at the individual level, but at the *aggregate* level. For instance, when Borge-Holthoefer and colleagues [10] use Twitter to study the polarization of sentiments during the 2013 Egyptian coup, they are not interested in the sentiments of the specific individual behind a specific Twitter account, but are interested in the aggregate data (possibly broken down according to various criteria) that can be extracted from the entire dataset under study: What is the fraction of tweeters who supported military intervention? What is the fraction of tweeters who supported Islamist groups? And how did these percentages evolve during the days of the coup? Similarly, when Dodds and colleagues [20] use Twitter in order to study the spatio-temporal patterns of happiness throughout the US population, they are not interested in how and when a specific person is happy, but are interested in the conclusions that the aggregate data allow them to draw. These examples are not isolated, and it is fair to say that most (if not all) TSC studies conducted, e.g., within political science [10,34,40], economics [9,49], social science [20], and market research [11,52], use Twitter with an interest in aggregate data and *not* in individual data.

Without loss of generality, we may say that TSC studies that focus on the aggregate level are concerned with estimating the *prevalence* (or "relative frequency") of each class of interest in the unlabelled dataset, i.e., with esti-

mating the distribution of the unlabelled data across the classes of interest. This task is known as *quantification* [4,7,24,27,43].

The obvious method for dealing with it is "classify and count", i.e., classifying each unlabelled object via a standard classifier and estimating class prevalence by counting the objects that have been labelled with the class. However, this strategy is suboptimal, since a good classifier is not necessarily a good "quantifier" (i.e., prevalence estimator). To see this consider that a binary classifier $h_1$ for which $FP = 20$ and $FN = 20$ ($FP$ and $FN$ standing for the "false positives" and "false negatives", respectively, it has generated on a given dataset) is worse than a classifier $h_2$ for which, on the same test set, $FP = 18$ and $FN = 20$. However, $h_1$ is intuitively a better binary quantifier than $h_2$; indeed, $h_1$ is a perfect quantifier, since $FP$ and $FN$ are equal and thus, when it comes to class frequency estimation, compensate each other, so that the distribution of the test items across the class and its complement is estimated perfectly. In other words, a good quantifier needs to have small *bias* (i.e., needs to distribute its errors as evenly as possible across $FP$ and $FN$).

Recent research (e.g., [4,7,24,27]) has convincingly shown that, since classification and quantification pursue different goals, quantification should be tackled as a task of its own, using different evaluation measures and, as a result, different learning algorithms. One reason why it seems sensible to pursue quantification directly, instead of tackling it via classification, is that classification is a more general task than quantification: after all, a perfect classifier is also a perfect quantifier, while the opposite is not true. A training set might thus contain information sufficient to generate a good quantifier but not a good classifier, which means that performing quantification via "classify and count" might be a suboptimal way of performing quantification. In other words, performing quantification via "classify and count" looks like a violation of "Vapnik's principle" [61], which asserts that "If you possess a restricted amount of information for solving some problem, try to solve the problem directly and never solve a more general problem as an intermediate step. It is possible that the available information is sufficient for a direct solution but is insufficient for solving a more general intermediate problem.".

In this paper we show, using 2 learners, 7 quantification-specific algorithms and 11 different TSC datasets, that quantification-specific algorithms indeed outperform, at prevalence estimation, state-of-the-art classification-oriented learning algorithms. We thus argue that researchers interested in tweet sentiment prevalence should switch to using quantification-specific (instead of classification-specific) learning algorithms and evaluation measures.

This paper is an extension of [28], where only experiments on one learner (instead of two), one quantification-specific algorithm (instead of seven), and eight TSC datasets (instead of eleven) were carried out; the conclusions that this much larger experimentation allow us to draw are thus more solidly grounded (and somehow surprising).

The paper is organized as follows. In Section 2 we discuss previous work in tweet sentiment classification and previous work in quantification, arguing that these two research streams have never crossed paths. In order to introduce

tweet sentiment quantification, in Section 3 we first look at the evaluation measures that are used in the quantification literature. In Section 4 we describe the tweet sentiment quantification systems we compare in this work; one system is based on "traditional" classification technology and other seven systems are based on quantification-specific learning algorithms. Section 5 describes the results of our experiments, while Section 6 concludes.

## 2 Related work

**Quantification methods.** Quantification goes under different names in different fields and different papers. It is variously called *prevalence estimation* [5], *counting* [38], *class probability re-estimation* [1], *class prior estimation* [12, 65], and *class distribution estimation* [29,39,64].

Different quantification methods have been proposed over the years, the two main classes being the *aggregative* and the *non-aggregative* methods. While the former require the classification of each individual item as an intermediate step, the latter do not, and estimate class prevalences holistically. Most methods (e.g., the ones described in [4,7,24,27,43]) fall in the former class (all the ones we use in this paper belong to this category), while the latter has few representatives (e.g., [29,35]).

Within the class of aggregative methods, a further distinction can be made between methods that use general-purpose learning algorithms (e.g., [7,27]), sometimes tweaking them or post-processing their prevalence estimates to account for their estimated bias, and methods that instead make use of learning algorithms explicitly devised for quantification (e.g., [4,24,43]). Aggregative quantification methods have a classifier under the hood; the underlying learning algorithms that have been used for learning such classifiers belong to the classes of kernel machines [3,4,24,27], decision trees [7,43], memory-based learning algorithms [7], logistic regression [3,7,37], and neural networks [6, 29]; most learning algorithms that have been used are of the cost-insensitive type, but cost-sensitive algorithms have been employed too [64]. Quantification has mostly been addressed at the binary level, although methods for single-label multi-class quantification [7,31,55] and ordinal quantification [17, 21] have been proposed.

A further distinction is that between batch learning methods, which require all the training examples to be loaded in memory at the same time, and incremental "online" methods, which relax this requirement and thus start learning after the first training examples are loaded in memory; practically all quantification methods studied up to now are batch methods, the only exception being the method discussed in [48].

**Applications of quantification.** Quantification has been applied to fields as diverse as epidemiology [35], remote sensing [37], marine ecology [6], resource allocation [27], word sense disambiguation [12], political science [31], and veterinary [29,57]. King and Lu [35] apply quantification to the estimation of cause-of-death prevalences from "verbal autopsies", i.e., verbal descriptions

of the symptoms suffered from deceased persons before dying. Latinne et al. [37] use a quantification approach in order to interpret automatically the land cover in images from remote sensing; similarly, Beijbom et al. [6] apply quantification to the task of surveying underwater surfaces that are covered by coral reef. Chan and Ng [12] use quantification in order to estimate word sense priors from a text dataset to disambiguate, so as to tune a word sense disambiguator to the estimated sense priors; their work can be seen as an instance of *transfer learning* (see e.g., [51]), since the goal is to adapt a word sense disambiguation algorithm to a domain different from the one the algorithm was trained upon. Hopkins and King [31] estimate the prevalence of support for different political candidates from blog posts, using the method pioneered in [35]. Forman [27] uses quantification for estimating the prevalence of different issues from logs of calls to customer support; these estimates allow a company to allocate more human resources to the issues which have elicited more calls. In [29] and [57] quantification is used for establishing the prevalence of damaged sperm cells in a given sample for veterinary applications. Saerens et al. [55] (followed in this by other authors [1,64,65]) instead apply quantification to customizing a trained classifier to the class prevalences of the test set, with the goal of improving classification accuracy on unlabelled data exhibiting a class distribution different from that of the training set. Balikas et al. [3] use quantification for model selection in supervised learning, i.e., they tune hyperparameters by choosing for them the values that yield the best quantification accuracy on the test data; this allows hyperparameter tuning to be performed without incurring the costs inherent in $k$-fold cross-validation.

**Shared tasks involving quantification.** Tweet quantification is one of the subjects of the recent SemEval Task 4 "Sentiment Analysis in Twitter" shared task [46], where tweets are labelled according to the sentiment they convey towards a certain topic. Subtask D consists of a *binary* quantification task, and Subtask E consists of an *ordinal* quantification task (with tweets labelled according to a five-point scale).

## 3 Evaluation measures for quantification

Let us look at the measures which are currently being used in the literature for evaluating quantification error; in Section 5 we will use the very same measures in evaluating the results of our experiments.

The task we tackle in this paper requires estimating the distribution of a set $\mathcal{S}$ of unlabelled tweets across a set $\mathcal{C}$ of available classes; we will typically deal with the case in which $|\mathcal{C}| = 3$, where the classes are Positive, Negative, and Neutral. Ours is thus a *single-label multi-class* (SLMC) quantification task, and we will thus concentrate on the measures that have been proposed for evaluating it. Note that a measure for SLMC quantification is also a measure for binary quantification, since the latter task is a special case of the former; this would be relevant for datasets in which the Neutral class is absent. Note also that a measure for binary quantification is also a measure for "multi-label

multi-class" quantification, since the latter task can be solved by separately solving $|\mathcal{C}|$ instances of the former task, one for each $c \in \mathcal{C}$.

Notation-wise, by $\Lambda(p, \hat{p}, S, \mathcal{C})$ we will indicate a *quantification loss*, i.e., a measure $\Lambda$ of the error made in estimating a distribution $p$ defined on set $\mathcal{S}$ and classes $\mathcal{C}$ by another distribution $\hat{p}$; we will often simply write $\Lambda(p, \hat{p})$ when $\mathcal{S}$ and $\mathcal{C}$ are clear from the context[1].

The simplest measure for SLMC quantification is *absolute error* (AE), which corresponds to the average (across the classes in $\mathcal{C}$) absolute difference between the predicted class prevalence and the true class prevalence; i.e.,

$$\mathrm{AE}(p, \hat{p}) = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} |\hat{p}(c) - p(c)| \tag{1}$$

It is easy to show that AE ranges between 0 (best) and

$$\frac{2(1 - \min_{c \in \mathcal{C}} p(c))}{|\mathcal{C}|}$$

(worst); a normalized version of AE that always ranges between 0 (best) and 1 (worst) can thus be obtained as

$$\mathrm{NAE}(p, \hat{p}) = \frac{\sum_{c \in \mathcal{C}} |\hat{p}(c) - p(c)|}{2(1 - \min_{c \in \mathcal{C}} p(c))} \tag{2}$$

The main advantage of AE and NAE is that they are intuitive, and easy to understand to non-initiates too.

However, AE and NAE do not address the fact that the same absolute difference between predicted class prevalence and true class prevalence should count as a more serious mistake when the true class prevalence is small. For instance, predicting $\hat{p}(c) = 0.10$ when $p(c) = 0.01$ and predicting $\hat{p}(c) = 0.50$ when $p(c) = 0.41$ are equivalent errors according to AE, but the former is intuitively a more serious error than the latter. *Relative absolute error* (RAE) addresses this problem by relativising the value $|\hat{p}(c) - p(c)|$ in Equation 1 to the true class prevalence, i.e.,

$$\mathrm{RAE}(p, \hat{p}) = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{|\hat{p}(c) - p(c)|}{p(c)} \tag{3}$$

RAE may be undefined in some cases, due to the presence of zero denominators. To solve this problem, in computing RAE we can smooth both $p(c)$ and $\hat{p}(c)$ via additive smoothing, i.e.,

$$p_s(c) = \frac{\epsilon + p(c)}{\epsilon|\mathcal{C}| + \sum_{c \in \mathcal{C}} p(c)} \tag{4}$$

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

where $p_s(c)$ denotes the smoothed version of $p(c)$ and the denominator is just a normalizing factor (same for the $\hat{p}_s(c)$'s); the quantity $\epsilon = \frac{1}{2|\mathcal{S}|}$ is often used as a smoothing factor. The smoothed versions of $p(c)$ and $\hat{p}(c)$ are then used in place of their original versions in Equation 3; as a result, RAE is always defined and still returns a value of 0 when $p$ and $\hat{p}$ coincide. It is easy to show that RAE ranges between 0 (best) and

$$\frac{|\mathcal{C}| - 1 + \frac{1 - \min_{c \in \mathcal{C}} p(c)}{\min_{c \in \mathcal{C}} p(c)}}{|\mathcal{C}|}$$

(worst); a normalized version of RAE that always ranges between 0 (best) and 1 (worst) can thus be obtained as

$$\mathrm{NRAE}(p, \hat{p}) = \frac{\sum_{c \in \mathcal{C}} \frac{|\hat{p}(c) - p(c)|}{p(c)}}{|\mathcal{C}| - 1 + \frac{1 - \min_{c \in \mathcal{C}} p(c)}{\min_{c \in \mathcal{C}} p(c)}} \tag{5}$$

A third measure, and the one that has become somehow standard in the evaluation of SLMC quantification, is *normalized cross-entropy*, better known as *Kullback-Leibler Divergence* (KLD – see e.g., [15]). KLD was proposed as a SLMC quantification measure in [26], and is defined as

$$\mathrm{KLD}(p, \hat{p}) = \sum_{c \in \mathcal{C}} p(c) \log_e \frac{p(c)}{\hat{p}(c)} \tag{6}$$

KLD was originally devised as a measure of the inefficiency incurred when estimating a true distribution $p$ over a set $\mathcal{C}$ of classes by means of a predicted distribution $\hat{p}$. KLD is thus suitable for evaluating quantification, since quantifying exactly means predicting how the items in set $\mathcal{S}$ are distributed across the classes in $\mathcal{C}$.

KLD ranges between 0 (best) and $+\infty$ (worst). Note that, unlike AE and RAE, the upper bound of KLD is not finite since Equation 6 has predicted probabilities, and not true probabilities, at the denominator: that is, by making a predicted probability $\hat{p}(c)$ infinitely small we can make KLD be infinitely large. A normalized version of KLD yielding values between 0 (best) and 1 (worst) may be defined by applying a logistic function[2], e.g.,

$$\mathrm{NKLD}(p, \hat{p}) = 2 \frac{e^{\mathrm{KLD}(p, \hat{p})}}{e^{\mathrm{KLD}(p, \hat{p})} + 1} - 1 \tag{7}$$

[2] Since the standard logistic function $\frac{e^x}{e^x+1}$ ranges (for the domain $[0, +\infty)$ we are interested in) on $[\frac{1}{2},1]$, we multiply by 2 in order for it to range on [1,2], and subtract 1 in order for it to range on [0,1], as desired.

Also KLD (and, as a consequence, NKLD) may be undefined in some cases. While the case in which $p(c) = 0$ is not problematic (since continuity arguments indicate that $0 \log \frac{0}{a}$ should be taken to be 0 for any $a \geq 0$), the case in which $\hat{p}(c) = 0$ and $p(c) > 0$ is indeed problematic, since $a \log \frac{a}{0}$ is undefined for $a > 0$. To solve this problem, also in computing KLD and NKLD we use the smoothed probabilities of Equation 4; as a result, KLD and NKLD are always defined and still return a value of zero when $p$ and $\hat{p}$ coincide.

While KLD is less easy to understand to non-initiates than AE or RAE, its advantage is that it is a very well-known measure, having been the subject of intense study within information theory [16] and, although from a more applicative angle, within the language modelling approach to information retrieval and to speech processing. As a consequence, it has emerged as the *de facto* standard in the SLMC quantification literature. We will thus pick it as the measure to optimize; however, in the experimental section we will report the results of all our experiments in terms of all six measures discussed above.

## 4 Tweet sentiment quantifiers

In this section we will describe the quantification systems we will use in our experiments. Like a classification system, a system for performing quantification consists of two main components: (i) an algorithm for converting the objects of interest (tweets, in our case) into vectorial representations that can be interpreted both by the learning algorithm and, once it has been trained, by the quantifier itself, and (ii) an algorithm for training quantifiers from vectorial representations of training objects. Section 4.1 describes component (i) while in Section 4.2 we describe the various choices for component (ii) that we have compared experimentally.

### 4.1 Features for detecting tweet sentiment

For building vectorial representations of tweets we have followed the approach discussed in [36, Section 5.2.1], since the representations presented therein are those used in the systems that performed best at both the SemEval 2013 [44] and SemEval 2014 [66] TSC shared tasks.

The text is preprocessed by normalizing URLs and mentions of users to the constants `http://someurl` and `@someuser`, resp., after which tokenisation and POS tagging is performed. The binary features used (i.e., features denoting presence or absence in the tweet) include word $n$-grams, for $n \in \{1, 2, 3, 4\}$, and character $n$-grams, for $n \in \{3, 4, 5\}$, whether the last token contains an exclamation and/or a question mark, whether the last token is a positive or a negative emoticon and, for each of the 1000 word clusters produced with the CMU Twitter NLP tool[3], whether any token from the cluster is present. Integer-valued features include the number of all-caps tokens, the number of

[3] `http://www.ark.cs.cmu.edu/TweetNLP/`

tokens for each POS tag, the number of hashtags, the number of negated contexts, the number of sequences of exclamation and/or question marks, and the number of elongated words (e.g., `cooooool`).

A key addition to the above is represented by features derived from both automatically generated and manually generated sentiment lexicons; for these features, we use the same sentiment lexicons as used in [36], which are all publicly available. We omit further details concerning our vectorial representations (and, in particular, how the sentiment lexicons contribute to them), both for brevity reasons and because these vectorial representations are not the central focus of this paper; the interested reader is invited to consult [36, Section 5.2.1] for details.

Finally, we should mention the fact that we did not perform any feature selection, since our learners could handle the resulting (huge) number of features fairly well from the standpoint of efficiency.

### 4.2 Learning to quantify

In order to test our conjecture that using quantification-specific algorithms, rather than standard classification-oriented ones, delivers superior quantification accuracy, in the experiments described in Section 5 we test seven quantification-specific algorithms against a state-of-the-art classification-oriented one. This section is devoted to describing each of them in detail.

Let us fix some notation. We assume a domain $D$ of objects; a generic object will be indicated by $\mathbf{x}$. We assume the availability of a set $Tr$ of training objects (tweets, in our case) and of a set $Te$ of test objects on which the accuracy of our quantifiers will be evaluated. A classifier (or *hypothesis*) trained on $Tr$ will be denoted by $h : D \rightarrow \mathcal{C}$, where $D$ is our domain of interest (the set of tweets) and $\mathcal{C}$ is the set of classes (Positive, Negative, Neutral, in our case). By $p_s(c)$ we will denote the true prevalence of class $c \in \mathcal{C}$ in set $s$, while by $\hat{p}_s^M(c)$ we will denote the prevalence of class $c \in \mathcal{C}$ in set $s$ as estimated via method $M$.

**Classify and Count (CC).** An obvious method for quantification consists of training a classifier from $Tr$ via a standard learning algorithm, classifying the objects in $Te$, and estimating $p_{Te}$ by simply counting the fraction of objects in $Te$ that are predicted to belong to the class. If by $\hat{c}$ we denote the event "class $c$ has been assigned by the classifier", so that $p_{Te}(\hat{c})$ represents the fraction of test documents that have been assigned $c$ by the classifier, this corresponds to computing

$$\begin{aligned} \hat{p}_{Te}^{CC}(c) &= p_{Te}(\hat{c}) \\ &= \frac{|\{\mathbf{x} \in Te | h(\mathbf{x}) = c\}|}{|Te|} \end{aligned} \tag{8}$$

Forman [27] calls this the *classify and count* (CC) method. This is the classification-oriented method we will use as a baseline in our experiments, and that will be contrasted with seven quantification-specific methods.

**Probabilistic Classify and Count (PCC).** A variant of CC consists in generating a classifier from $Tr$, classifying the objects in $Te$, and computing $p_{Te}(c)$ as the *expected* fraction of objects predicted to belong to $c$. If by $p(c|\mathbf{x})$ we indicate the posterior probability, i.e., the probability of membership in $c$ of test object $\mathbf{x}$ as estimated by the classifier, and by $E[x]$ we indicate the expected value of $x$, this corresponds to computing

$$\begin{aligned}\hat{p}_{Te}^{PCC}(c) &= E[p_{Te}(\hat{c})] \\ &= \frac{1}{|Te|}\sum_{\mathbf{x}\in Te} p(c|\mathbf{x})\end{aligned} \tag{9}$$

The rationale of PCC is that posterior probabilities contain richer information than binary decisions, which are usually obtained from posterior probabilities by thresholding.

If the classifier only returns confidence scores that are not probabilities (as is the case, e.g., when the scores do no range on [0,1]), the former must be converted into true probabilities. If the score is a monotonically increasing function of the classifier's confidence in the fact that the object belongs to the class, this may be obtained by applying a logistic function. *Well-calibrated* probabilities (defined as the probabilities such that the prevalence $p_s(c)$ of a class $c$ in a set $s$ is equal to $\sum_{\mathbf{x}\in s} p(c|\mathbf{x})$) may be obtained by using a *generalized* logistic function; see e.g., [8, Section 4.4] for details.

The PCC method is dismissed as unsuitable in [26,27], on the grounds that, when the training set distribution $p_{Tr}$ and the test set distribution $p_{Te}$ are different (as they should be assumed to be in any application of quantification), probabilities calibrated on $Tr$ ($Tr$ being the only available set where calibration may be carried out) cannot be, by definition, calibrated for $Te$ at the same time. Experimental evidence on PCC is not conclusive, since PCC performed better than CC in the experiments of [7] (where it is called "Probability Average") and [59], but underperformed CC in the (much more extensive) experiments of [24].

**Adjusted Classify and Count (ACC).** Forman [26,27] uses a further method which he calls "Adjusted Count", and which we will call (consistently with [24]) *Adjusted Classify and Count* (ACC) so as to make its relation with CC more explicit.

ACC is based on the observation that, thanks to the law of total probability, it holds that

$$p_{Te}(\hat{c}_j) = \sum_{c_i\in\mathcal{C}} p_{Te}(\hat{c}_j|c_i)\cdot p_{Te}(c_i) \tag{10}$$

Here, $p_{Te}(\hat{c}_j|c_i)$ represents the fraction of test documents belonging to $c_i$ that have been instead assigned $c$ by the classifier. Note that, once the classifier has been trained and applied to $Te$, the quantity $p_{Te}(\hat{c}_j)$ can be observed, and the quantity $p_{Te}(\hat{c}_j|c_i)$ can be estimated from $Tr$ via $k$-fold cross-validation; the quantity $p_{Te}(c_i)$ is instead unknown, and is indeed the quantity we want to estimate. Since there are $|\mathcal{C}|$ equations of the type described in Equation 10 (one for each possible $\hat{c}_j$), and since there are $|\mathcal{C}|$ quantities of type $p_{Te}(c_i)$

to estimate (one for each choice of $c_i$), we are in the presence of a system of $|\mathcal{C}|$ linear equations in $|\mathcal{C}|$ unknowns. This system can be solved via standard techniques, thus yielding the required $\hat{p}_{Te}(c_i)$ estimates.

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 $p_{Te}(\hat{c}_j|c_i)$ may be imperfect. This has lead most authors (see e.g., [27]) to (i) "clip" the $\hat{p}_{Te}(c_i)$ estimates (i.e., equate to 1 every value higher than 1 and to 0 every value lower than 0), and (ii) rescale them so that they sum up to 1.

**Probabilistic Adjusted Classify and Count (PACC).** The PACC method (proposed in [7], where it is called "Scaled Probability Average") is a probabilistic variant of ACC, i.e., it stands to ACC like PCC stands to CC. Its underlying idea is to replace, in Equation 10, $p_{Te}(\hat{c}_j)$ and $p_{Te}(\hat{c}_j|c_i)$ with their expected values. Equation 10 is thus transformed into

$$E[p_{Te}(\hat{c}_j)] = \sum_{c_i \in \mathcal{C}} E[p_{Te}(\hat{c}_j|c_i)] \cdot p_{Te}(c_i) \tag{11}$$

where

$$\begin{aligned} E[p_{Te}(\hat{c}_j)] &= \frac{1}{|Te|} \sum_{\mathbf{x} \in Te} p(c_j|\mathbf{x}) \\ E[p_{Te}(\hat{c}_j|c_i)] &= \frac{1}{|Te_i|} \sum_{\mathbf{x} \in Te_i} p(c_j|\mathbf{x}) \end{aligned} \tag{12}$$

and $Te_i$ indicates the set of objects in $Te$ whose true class is $c_i$. Like for ACC, once the classifier has been trained and applied to $Te$, the quantity $E[p_{Te}(\hat{c}_j)]$ can be observed, and the quantity $E[p_{Te}(\hat{c}_j|c_i)]$ can be estimated from $Tr$ via $k$-fold cross-validation, which means that we are again in the presence of a system of $|\mathcal{C}|$ linear equations in $|\mathcal{C}|$ unknowns, that we can solve by standard techniques. Like ACC, also PACC can return $\hat{p}_{Te}(c_i)$, i.e., estimates of $p_{Te}(c_i)$, that fall off the [0,1] range; again, clipping and rescaling is the only solution in these cases.

Like PCC, also PACC is dismissed as unsuitable in [26, 27], for the same reasons for which PCC was also dismissed. Unlike in the case of PCC, published experimental evidence seems instead in favour of PACC, since the experimental results published in [7, 24, 59] indicate PACC to outperform all of CC, PCC, and ACC.

**Expectation Maximization for Quantification (EMQ).** EMQ, proposed by Saerens et al. [55], is an instance of Expectation Maximization [18], a well-known iterative algorithm for finding maximum-likelihood estimates of parameters (in our case: the class prevalences) for models that depend on unobserved variables (in our case: the class labels). Essentially, EMQ (see Algorithm 1) incrementally updates (Line 10) the posterior probabilities by using the class prevalences computed in the last step of the iteration, and updates (Line 14) the class prevalences by using the posterior probabilities computed in the last step of the iteration, in a mutually recursive fashion.

**SVMs optimized for** KLD **(SVM(KLD)).** SVM(KLD), proposed in [22, 24], is an instantiation of Thorsten Joachims' SVM-perf [32] that uses

**Input** : Class prevalences $p_{Tr}(c)$ on $Tr$, for all $c \in \mathcal{C}$;
Posterior probabilities $p(c|\mathbf{x})$, for all $c \in \mathcal{C}$ and for all $\mathbf{x} \in Te$;
**Output**: Estimates $\hat{p}_{Te}(c)$ of class prevalences on $Te$;

`/* Initialization */`

$s \leftarrow 0$;
**for** $c \in \mathcal{C}$ **do**
  $\hat{p}_{Te}^{(s)}(c) \leftarrow p_{Tr}(c)$;
  **for** $\mathbf{x} \in Te$ **do**
    $p^{(s)}(c|\mathbf{x}) \leftarrow p(c|\mathbf{x})$;
  **end**
**end**

`/* Main Iteration Cycle */`

**while** *stopping condition = false* **do**
  $s \leftarrow s + 1$;
  **for** $c \in \mathcal{C}$ **do**
    **for** $\mathbf{x} \in Te$ **do**

$$p^{(s)}(c|\mathbf{x}) \leftarrow \frac{\frac{\hat{p}_{Te}^{(s)}(c)}{\hat{p}_{Te}^{(0)}(c)} \cdot p^{(0)}(c|\mathbf{x})}{\sum_{c \in \mathcal{C}} \frac{\hat{p}_{Te}^{(s)}(c)}{\hat{p}_{Te}^{(0)}(c)} \cdot p^{(0)}(c|\mathbf{x})}$$

    **end**
    $\hat{p}_{Te}^{(s)}(c) \leftarrow \frac{1}{|Te|} \sum_{\mathbf{x} \in Te} p^{(s-1)}(c|\mathbf{x})$
  **end**
**end**

`/* Generate output */`

**for** $c \in \mathcal{C}$ **do**
  $\hat{p}_{Te}(c) \leftarrow \hat{p}_{Te}^{(s)}(c)$
**end**

**Algorithm 1:** The EMQ algorithm [55].

KLD as the loss to optimize[4]. SVM-perf is a "structured output prediction" algorithm in the support vector machines (SVMs) family. Unlike traditional SVMs, SVM-perf is capable of optimizing any nonlinear, multivariate loss function that can be computed from a contingency table (as all the measures presented in Section 3 are). Instead of handling hypotheses $h : \mathcal{X} \rightarrow \mathcal{Y}$ that map an individual item (in our case: a tweet) $\mathbf{x}_i$ into an individual label $y_i$, SVM-perf considers hypotheses $\bar{h} : \bar{\mathcal{X}} \rightarrow \bar{\mathcal{Y}}$ that map entire tuples of items (in our case: entire sets of tweets) $\bar{\mathbf{x}} = (\mathbf{x}_1, ..., \mathbf{x}_n)$ into tuples of labels $\bar{\mathbf{y}} = (y_1, ..., y_n)$. Instead of learning the traditional hypotheses of type

$$h(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x} + b) \tag{13}$$

SVM-perf thus learns hypotheses of type

$$\bar{h}(\bar{\mathbf{x}}) = \arg\max_{\bar{y} \in \bar{\mathcal{Y}}} (\mathbf{w} \cdot \Psi(\bar{\mathbf{x}}, \bar{y})) \tag{14}$$

[4] In [32] SVM-perf is actually called SVM-multi, but the author has released its implementation under the name SVM-perf; we will thus use this latter name.

where $\mathbf{w}$ is the vector of parameters to be learnt during training and

$$\Psi(\bar{\mathbf{x}}, \bar{y}) = \sum_{i=1}^{n} \mathbf{x}_i y_i \tag{15}$$

(the *joint feature map*) is a function that scores the pair of tuples $(\bar{\mathbf{x}}, \bar{y})$ according to how "compatible" $\bar{\mathbf{x}}$ and $\bar{y}$ are. In other words, while classifiers trained via traditional SVMs classify individual instances $\mathbf{x}$ one at a time, models trained via SVM-perf classify entire sets $\bar{\mathbf{x}}$ of instances in one shot, and can thus make the labels assigned to the individual items mutually depend on each other. This is of fundamental importance in quantification, where, say, an additional false positive may even be *beneficial* when the rest of the data is expected to contain more false negatives than false positives.

While the optimization problem of classic soft-margin SVMs consists of finding

$$\begin{array}{ll} \arg\min\limits_{\mathbf{w},\xi_i \geq 0} & \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum\limits_{i=1}^{|Tr|}\xi_i \\ \text{such that} & y'_i[\mathbf{w}\cdot\mathbf{x}'_i + b] \geq (1-\xi_i) \\ & \qquad \text{for all } i \in \{1,...,|Tr|\} \end{array} \tag{16}$$

(where the $(\mathbf{x}'_i, y'_i)$ denote the training examples), the corresponding problem of SVM-perf consists instead of finding

$$\begin{array}{ll} \arg\min\limits_{\mathbf{w},\xi \geq 0} & \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\xi \\ \text{such that} & \mathbf{w}\cdot[\Psi(\bar{\mathbf{x}}', \bar{y}') - \Psi(\bar{\mathbf{x}}', \bar{y}) + b] \\ & \qquad \geq \Lambda(\bar{y}', \bar{y}) - \xi \text{ for all } \bar{y} \in \bar{\mathcal{Y}}/\bar{y}' \end{array} \tag{17}$$

where $(\bar{\mathbf{x}}', \bar{y}')$ indicates a sequence of training examples and the corresponding sequence of their true labels. Here, the relevant fact to observe is that the multivariate loss $\Lambda$ explicitly appears in the optimization problem.

We refer the interested reader to [32,33,60] for more details on SVM-perf (and on SVMs for structured output prediction in general). From the point of view of the user interested in applying it to a certain task, the implementation of SVM-perf made available by its author is essentially an off-the-shelf package, since for customizing it to a specific loss $\Lambda$ one only needs to write a small module that describes how to compute $\Lambda$ from a contingency table.

**SVMs optimized for** NKLD **(SVM(NKLD)).** SVM(NKLD), originally discussed in [23], is just a minor variation of SVM(KLD), the only difference being that NKLD (as from Equation 7), is the loss function being minimized.

**SVMs optimized for** $Q$ **(SVM(Q)).** SVM(Q), originally proposed in [4], is (like SVM(KLD) and SVM(NKLD)) an instantiation of SVM-perf. The authors optimize a "multi-objective" measure (which they call *Q-measure*) that combines classification accuracy and quantification accuracy; the rationale is that by maximizing both measures at the same time, one tends to obtain quantifiers that are not just effective (thanks to the high quantification

accuracy), but also reliable (thanks to the high classification accuracy). The authors' Q-measure is

$$Q_\beta(p,\hat{p}) = \frac{(\beta^2+1)\Gamma_c(p,\hat{p})\cdot\Gamma_q(p,\hat{p})}{\beta^2\Gamma_c(p,\hat{p})+\Gamma_q(p,\hat{p})} \tag{18}$$

where $\Gamma_c$ and $\Gamma_q$ are a measure of classification "gain" (the opposite of loss) and a measure of quantification gain, respectively, and $0 \leq \beta \leq +\infty$ is a parameter that controls the relative importance of the two; for $\beta = 0$ the $Q_\beta$ measure coincides with $\Gamma_c$, while when $\beta$ tends to $+\infty$, $Q_\beta$ asymptotically tends to $\Gamma_q$.

As a measure of classification gain the authors use recall, while as a measure of quantification gain they use $(1-\text{NAE})$, where NAE is as defined in Equation 2. The authors motivate the (apparently strange) decision to use recall as a measure of classification gain with the fact that, while recall by itself is not a suitable measure of classification gain (since it is always possible to arbitrarily increase recall at the expense of precision or specificity), to include precision or specificity in $Q_\beta$ is unnecessary, since the presence of $\Gamma_q$ in $Q_\beta$ has the effect of ruling out anyway those hypotheses characterised by high recall and low precision / specificity (since these hypotheses are indeed penalized by $\Gamma_q$)[5].

## 5 Experiments

### 5.1 Datasets

We have carried out our experiments on a variety of TSC datasets previously used in the literature; the main characteristics of these datasets are listed in Table 1. The SemEval2013, SemEval2014, SemEval2015, and SemEval2016 datasets are described more in detail in [47], [54], [53], and [46], respectively, while all of the other datasets (Sanders, SST, OMD, HCR, GASP, WA, WB) are described in detail in [56]. Our choice of datasets has followed two main guidelines, i.e., (i) selecting publicly available datasets, so as to guarantee a high level of replicability, and (ii) selecting datasets whose sentiment labels are the result of manual annotation, so as to guarantee high label quality[6].

It is well known that, when Twitter data are concerned, the replicability of experimental results is limited since, due to terms of use imposed by Twitter, the datasets made available by researchers cannot contain the tweets themselves, but only consists of their id's; the tweets corresponding to some of the id's may become unavailable over time, which means that the datasets we use

[5] SVM-perf is available from `http://svmlight.joachims.org/svm_struct.html`, while the module that customizes it to KLD is available from `http://hlt.isti.cnr.it/quantification/` . The code for all the other methods discussed in this section is available from `http://alt.qcri.org/~wgao/codes/tweet_sentiment_quantification.zip`.

[6] This means that we avoid STC datasets in which the labels are automatically derived from, say, the emoticons present in the tweets.

**Table 1** Datasets used in this work and their main characteristics. The last column indicates *distribution drift* measured in terms of KLD($p_{Te}, p_{Tr}$), i.e., indicates how much the distribution of the data across the three classes in the test set diverges from that in the training set, with higher values indicating higher divergence. The datasets are listed in increasing order of their KLD($p_{Te}, p_{Tr}$) value, and grouped into low-drift (LD), medium-drift (MD), and high-drift ones (HD); using any of AE, NAE, NKLD in place of KLD would have generated the same ranking.

| | Dataset | # of features used | # of training tweets | # of held-out tweets | # of test tweets | Total # of tweets | Distribution drift |
|---|---|---|---|---|---|---|---|
| LD | Sanders | 229,399 | 1,847 | 308 | 923 | 3,078 | 0.000027 |
| | WB | 404,333 | 3,650 | 609 | 1,823 | 6,082 | 0.000035 |
| | SemEval2016 | 889,504 | 6,000 | 2,000 | 2,000 | 10,000 | 0.000289 |
| | GASP | 694,582 | 7,532 | 1,256 | 3,765 | 12,553 | 0.000434 |
| MD | OMD | 199,151 | 1,576 | 263 | 787 | 2,626 | 0.001344 |
| | WA | 248,563 | 1,872 | 312 | 936 | 3,120 | 0.002127 |
| | SemEval2013 | 1,215,742 | 9,684 | 1,654 | 3,813 | 15,151 | 0.003827 |
| HD | SST | 376,132 | 2,546 | 425 | 1,271 | 4,242 | 0.008259 |
| | SemEval2015 | 1,215,742 | 9,684 | 1,654 | 2,390 | 13,728 | 0.008566 |
| | HCR | 222,046 | 797 | 797 | 798 | 2,392 | 0.018663 |
| | SemEval2014 | 1,215,742 | 9,684 | 1,654 | 1,853 | 13,191 | 0.051052 |

here are typically subsets of the original datasets. Luckily enough, this problem affects us only marginally since (i) of the four SemEval datasets we owned an original copy before starting this research, and (ii) we were able to recover all of the original tweets in all of the other datasets (except for Sanders).

Most of the above datasets classify tweets across the three classes Positive, Negative, Neutral; some others (Sanders, OMD, HCR, GASP, WA, WB) also use additional classes (e.g., Mixed, Irrelevant, Other), and SST uses 10 different levels of sentiment strength (from VeryPositive to VeryNegative). For reasons of uniformity, we have removed the tweets belonging to the additional classes (Sanders, OMD, HCR, GASP, WA, WB), and converted sentiment strengths into Positive, Negative, Neutral using the same heuristics as described in [56] (SST); in all of the datasets we use, the task is thus to quantify the Positive, Negative, Neutral classes, which represent a partition of the dataset.

Because of the reasons above, the numbers reported in Table 1 refer not to the original datasets but to the versions we have used[7].

Table 1 lists our eleven datasets in increasing order of their *distribution drift*, i.e., of how differently the training set and the test set are distributed across the three classes of interest. Distribution drift can be measured by any

[7] In order to enhance the reproducibility of our experimental results, we make available (at `http://alt.qcri.org/~wgao/data/SNAM/tweet_sentiment_quantification.zip`) the vectorial representations we have generated for all the datasets (split into training / validation / test sets) used in this paper.

of the six evaluation measures we have discussed in Section 3. It turns out that AE, NAE, KLD, NKLD, return exactly the same ranking of our eleven datasets, while RAE and NRAE each return a slightly different ranking. In Table 1 we thus use the ranking returned by AE, NAE, KLD (for which we also report actual values), NKLD, since it is the one most measures agree upon.

For better convenience, we have broken down the eleven datasets into three groups, based on distribution drift:

1. Sanders, WB, Semeval2016, GASP, are the *low-drift* (LD) datasets, characterised by values $\text{KLD}(p_{Te}, p_{Tr}) < 0.0005$; the results obtained on these datasets are reported in Table 5;
2. OMD, WA, Semeval2013, are the *medium-drift* (MD) datasets, characterised by values $0.0005 < \text{KLD}(p_{Te}, p_{Tr}) < 0.0050$; the results obtained on these datasets are reported in Table 6;
3. SST, Semeval2015, HCR, Semeval2014, are the *high-drift* (HD) datasets, characterised by values $\text{KLD}(p_{Te}, p_{Tr}) > 0.0050$; the results obtained on these datasets are reported in Table 7.

In order to allow better insight into the behaviour of our quantification methods, we will break down the experimental results we obtain according to these three groups.

### 5.2 Experiments with SVMs as a base learner

In the experiments we describe in this section, on each dataset we compare the quantification-specific learning algorithms of Section 4.2 (PCC, ACC, PACC, EMQ, SVM(KLD), SVM(NKLD), SVM(Q)) against a baseline (CC) consisting of a representative, state-of-the-art, classification-specific learning algorithm. For the CC baseline we use a standard SVM with a linear kernel, in the implementation made available in the LIBSVM system[8] [13]; it is a strong baseline, and is (among others) the learning algorithm used in the systems that performed best at both the SemEval 2013 [44] and SemEval 2014 [66] STC shared tasks. While SVM(KLD), SVM(NKLD), and SVM(Q) explicitly minimize KLD, NKLD, and Q-measure, resp., the above baseline minimizes the well-known Hinge Loss.

All 8 quantification methods use the same vectorial representations, as described in Section 4.1. All the learning algorithms we discuss in this section are based on SVMs. The rationale of this choice was (i) to obtain high accuracy throughout the spectrum of the 8 quantification methods tested, given that SVMs are known to deliver high accuracy across different text mining tasks, and (ii) to allow a fair comparison between (a) SVM(KLD), SVM(NKLD), SVM(Q), which are *inherently* based on SVM technology, and (b) CC, PCC, ACC, PACC, EMQ, which could in principle have relied on other learning technologies (e.g., a naïve Bayesian classifier).

[8] The SVM-based implementation of CC is called SVM(HL) in [28]. LIBSVM is available from `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

For PCC, ACC, PACC, EMQ, we use the same output that was generated by the LIBSVM-based classifier discussed in the previous paragraph, and that is used for the CC baseline. Note that PCC, PACC, EMQ, do not require as output the classifier's binary decisions, but the posterior probabilities generated by the classifier. Since SVMs do not natively generate posterior probabilities, we use the `-b` option of LIBSVM, which converts the scores originally generated by SVMs into posterior probabilities according to the algorithm described in [63].

For each of the 8 quantification methods we have optimized the $C$ parameter (which sets the tradeoff between the training error and the margin – see Equations 16 and 17) via validation on a separate held-out set, performing a grid search on all values of type $10^x$ with $x \in \{-6, ..., 7\}$; we have optimized $C$ individually for each (method, dataset) pair. We have instead left the other parameters at their default value; in particular, we have used a linear kernel. For the EMQ algorithm, we stop the iteration when $\sum_{c \in \mathcal{C}} |\hat{p}_{Te}^{(s)}(c) - \hat{p}_{Te}^{(s-1)}(c)| < 10^{-10}$. For the SVM(Q) method we have set the $\beta$ parameter (see Equation 18) to 2, as recommended in [4]. Some of the datasets we use (SemEval2013, SemEval2014, SemEval2015, SemEval2016, and HCR) already come with a predefined split between training set and held-out set, with (for the SemEval2013, SemEval2014, SemEval2015 datasets) roughly six times as many training items as held-out items[9]; for the datasets where such split is not predefined, we have randomly selected the held-out examples from the training examples, using the same ratio as in the SemEval2013, SemEval2014, SemEval2015 datasets. For all datasets, after the optimal parameter values have been selected we have retrained the classifier on the union of the training and the held-out sets.

As noted in Section 3, ours is a single-label multi-class task. This does not pose any problem to our baseline system, since LIBSVM is equipped with a built-in SLMC option; this ensures that the baseline is a strong one. It instead poses a problem to SVM(KLD), SVM(NKLD), SVM(Q), which are binary learning algorithms. We circumvent this problem by (i) using each of SVM(KLD), SVM(NKLD), SVM(Q) to train $|\mathcal{C}|$ "one-against-all" binary predictors, (ii) having each binary predictor output a prevalence estimate for the corresponding class, and (iii) normalizing these prevalence estimates so that they sum up to 1. For these three methods, the optimization of the $C$ parameter mentioned in the previous paragraph is carried out individually for each "one-against-all" binary predictor.

The overall results of our SVM-based experiments are reported in Table 2; for each quantification method we present (a) the score (averaged across the eleven datasets) obtained by the method according to each of the six evaluation measures we consider, and (b) the relative deterioration in accuracy brought about by each method with respect to the best-performing system.

[9] At the time of writing this paper, the test set of the SemEval2016 collection has not yet been made available. However, the data made available by the organizers was already pre-split into three subsets, called "train", "dev", and "devtest"; we have thus used these subsets as the training set, held-out set, and test set, respectively.

**Table 2** Quantification accuracy averaged over all eleven STC datasets, obtained with a classification-oriented learning algorithm (CC) and seven quantification-oriented learning algorithms (PCC, ACC, PACC, EMQ, SVM(KLD), SVM(NKLD), SVM(Q)), using SVMs as the base learner. The algorithms are ranked in increasing order of the KLD value they have obtained. Percentages represent deteriorations with respect to the best-performing system, indicated in **boldface**.

| System | AE | +% | NAE | +% | RAE | +% | NRAE | +% | KLD | +% | NKLD | +% |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PCC | **0.0291** | | **0.0528** | | **0.0949** | | **0.0406** | | **0.0068** | | **0.0034** | |
| ACC | 0.0339 | (+16.4%) | 0.0611 | (+15.7%) | 0.1128 | (+18.9%) | 0.0468 | (+15.1%) | 0.0126 | (+84.0%) | 0.0063 | (+84.0%) |
| EMQ | 0.0422 | (+44.8%) | 0.0779 | (+47.4%) | 0.1449 | (+52.7%) | 0.0651 | (+60.3%) | 0.0146 | (+113.5%) | 0.0073 | (+113.5%) |
| PACC | 0.0361 | (+23.8%) | 0.0653 | (+23.5%) | 0.1315 | (+38.6%) | 0.0546 | (+34.4%) | 0.0156 | (+128.4%) | 0.0078 | (+128.4%) |
| SVM(KLD) | 0.0530 | (+81.9%) | 0.0979 | (+85.3%) | 0.1713 | (+80.5%) | 0.0786 | (+93.4%) | 0.0216 | (+216.4%) | 0.0108 | (+216.4%) |
| CC | 0.0566 | (+94.1%) | 0.1027 | (+94.4%) | 0.1799 | (+89.6%) | 0.0765 | (+88.2%) | 0.0244 | (+257.1%) | 0.0122 | (+257.1%) |
| SVM(NKLD) | 0.0700 | (+140.3%) | 0.1276 | (+141.5%) | 0.2790 | (+193.9%) | 0.1167 | (+187.3%) | 0.0351 | (+413.4%) | 0.0175 | (+413.3%) |
| SVM(Q) | 0.0874 | (+199.9%) | 0.1591 | (+201.1%) | 0.3365 | (+254.5%) | 0.1400 | (+244.5%) | 0.0474 | (+593.3%) | 0.0237 | (+593.0%) |

In Tables 5 to 7 we report the results obtained by our methods on each individual dataset, where the datasets are clustered into the three groups identified in Section 5.1. In each such table, aside from the results obtained by each method for each of the six considered evaluation measures (Columns 6–11), we report the class distribution computed for each dataset by each method (Columns 3–5), which can be easily compared with the actual class distributions in the training set and in the test set.

Let us start from discussing the overall results in Table 2. A first observation we can make is that the six measures we use are in substantial agreement over which method is better than which other. In Table 2 the methods are ranked by their KLD value, but ranking them by any other measure would not have changed the ranking much. In fact, AE, NAE, RAE, NRAE only disagree with KLD, NKLD on which between EMQ and PACC is better, and NRAE disagrees with all others on which between SVM(KLD) and CC is better; however, all six measures agree everywhere else.

If we want to compare the learning methods with each other, Table 2 clearly says that PCC is the best method, according to all six measures. Since the key to PCC's performance is the accuracy of the posterior probabilities it receives as input, this is an indirect indication that the method of [63], which is used in LIBSVM to map confidence scores into posterior probabilities, is an effective one. Note also that EMQ is substantially outperformed by PCC, which coincides with EMQ when this latter is stopped after one iteration only; it thus looks like iterations beyond the first, which were carried out in order to obtain convergence, were actually detrimental, instead of beneficial, to the accuracy of EMQ.

The success of PCC might come as a surprise in the light of the results of [24], whose extensive experiments had shown SVM(KLD) to clearly outperform PCC (and other methods tested here such as CC, ACC, PACC). However, a closer look at the results of [24] shows that SVM(KLD) was the best quantifier only on low-prevalence classes ($p_{Te}(c) < 0.01$) or mid-prevalence classes ($0.01 \leq p_{Te}(c) < 0.10$), but was not on high-prevalence classes ($p_{Te}(c) \geq 0.10$), where it was outperformed (among others) by PACC and ACC; and all 3

classes in all our 11 datasets (with the only exception of Positive in GASP) are indeed high-prevalence classes. This is interesting in view of the fact that the task analysed in [24] (multi-label classification by topic) and the one we analyse here (sentiment classification) are quite different when it comes to class distributions. Multi-label classification by topic generally uses a large set of classes (often in the form of a complex taxonomy), and these large sets usually exhibit a power-law behaviour, with very few high-prevalence classes and very many mid- or low-prevalence classes; in these settings, as shown in [24], SVM(KLD) seems to shine. Instead, sentiment classification generally uses few classes (usually: two or three), which are usually high-prevalence, and the present paper seems to suggest that SVM(KLD) is suboptimal in these contexts. While this certainly deserves further investigation, we may hypothesise that, while SVM(KLD) might be the best choice for multi-label classification by topic, it is instead not in sentiment classification, which is characterised by radically different class distributions. The present work seems to indicate that, in these contexts, PCC is the method of choice.

If we look at the results on the individual datasets (see Tables 5 to 7), the superiority of PCC seems even more blatant. Out of 66 combinations of 11 datasets $\times$ 6 evaluation measures, PCC is the best performer in 34, while ACC is in 18, PACC is in 8, and EMQ is in 6. If we look at the average results on low-drift and mid-drift datasets, PCC is always the best performer; on high-drift datasets, PCC is the best according to two measures, ACC is the best according to three measures, while EMQ prevails according to one measure.

These latter figures highlight the (somehow surprising) absence, from the lot of the best performers, of the three methods based on structured prediction (SVM(KLD), SVM(NKLD), SVM(Q)), which would seem the better motivated ones from a theoretical point of view. These three methods are here the worst among the quantification-specific algorithms, with SVM(NKLD), SVM(Q) even beaten by standard CC. One possible explanation might be the fact that, as mentioned in Section 5.2, unlike the other methods these three methods are binary in nature (since there is no known multi-class equivalent of the SVM-perf method they are based upon), which means that the ternary quantification task we tackle in this paper must be accomplished by (a) generating independent individual estimates of class prevalence for each of the three classes of interest, by running three independent binary quantifiers, and then (b) normalizing the resulting estimates so that they sum up to 1. This is quite different, and somehow suboptimal, from methods that (as evident by looking, say, at Equations 10 and 11, and at Algorithm 1) are "natively multi-class".

We also remark that the results reported in Table 2 by and large confirm the results we had reported in the earlier version of this paper [28], where only CC and SVM(KLD) had been tested. Like in [28], where we experimented only on 8 out of the present 11 datasets, SVM(KLD) outperforms CC. Here, SVM(KLD) outperforms CC according to five out of six measures (NRAE being the exception), while it was superior according to all six measures in [28].

Last but not least, we observe that CC, our classification-specific baseline, is outperformed by all quantification-specific learning methods (except SVM(NKLD) and SVM(Q)), and that the difference in performance is very substantial. In fact, the increase in quantification error brought about by using CC instead of the best-performing quantification-specific method (PCC) ranges between +88.2% (when quantification error is measured by NRAE) and +257.1% (when it is measured by KLD). This confirms the basic claim of this paper, i.s., that when tweet sentiment classification is carried out with the only goal of estimating the prevalence of the classes of interest, we should use a quantification-specific method rather than a generic classifier.

### 5.3 Experiments with L2-regularized logistic regression as a base learner

While the last three algorithms of Section 4.2 (SVM(KLD), SVM(NKLD), SVM(Q)) are inherently based on SVMs, the first five (CC, PCC, ACC, PACC, EMQ) can be used in connection with any learner. (Technically, PCC, PACC, EMQ require the learner to return a posterior probability, but this can be obtained from any learner that, for an unlabelled document, outputs a numerical confidence score instead of just a binary decision; see the discussion of PCC in Section 4.2.)

As a result, we have performed further experiments in which we test CC, PCC, ACC, PACC, EMQ also in connection with a learner different from SVMs. As the learner, we have chosen L2-regularized logistic regression (L2-LR), as available in the LIBLINEAR package[10][25]. The reasons why we have chosen L2-LR are that (a) it has consistently delivered state-of-the-art performance in several applications [14, 67], and is thus a strong contender, and (b) logistic regression, unlike SVMs, natively outputs posterior probabilities (see e.g., [45, Section 1.4.6]), which is helpful.

For the benefit of PCC, PACC, EMQ, which require the base classifier to generate posterior probabilities, we use the `-b` option of LIBLINEAR, which converts the scores originally generated by L2-LR into posterior probabilities. L2-LR has only one parameter, the $C$ parameter which sets the amount of regularization to be applied; we have optimized this parameter in the same way as we did for SVMs' $C$ parameter (see Section 5.2). For the EMQ algorithm, we stop the iteration in the way described in Section 5.2.

The overall results of these further experiments are reported in Table 3, which reports for L2-LR the same data that Table 2 had reported for SVMs. A comparison between Table 2 and Table 3 shows that the L2-LR experiments substantially confirm the conclusions we had drawn from the SVM experiments. The PCC method is still the best of the lot, with a very substantial margin over the second best method, which is again ACC. Both the SVM and the L2-LR experiments, and all of the six tested evaluation measures, indicate that PCC > ACC > PACC > CC (where ">" stands for "is a better method

[10] `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

**Table 3** As Table 2, but with L2-LR as the base learner in place of SVMs. The algorithms are ranked in increasing order of the KLD value they have obtained. Percentages represent deteriorations with respect to the best-performing system, indicated in **boldface**.

| System | AE | +% | NAE | +% | RAE | +% | NRAE | +% | KLD | +% | NKLD | +% |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PCC | **0.0298** | | **0.0536** | | **0.1213** | | **0.0485** | | **0.0099** | | **0.0049** | |
| ACC | 0.0463 | (+55.3%) | 0.0844 | (+57.5%) | 0.1617 | (+33.3%) | 0.0693 | (+43.0%) | 0.0161 | (+63.1%) | 0.00.81 | (+63.1%) |
| PACC | 0.0484 | (+162.4%) | 0.0878 | (+163.8%) | 0.1687 | (+139.2%) | 0.0712 | (+146.9%) | 0.0212 | (+113.7%) | 0.0106 | (+113.7%) |
| CC | 0.0586 | (+196.7%) | 0.1063 | (+198.3%) | 0.1926 | (+158.9%) | 0.0813 | (+167.7%) | 0.0240 | (+142.3%) | 0.0120 | (+142.3%) |
| EMQ | 0.0667 | (+224.0%) | 0.1216 | (+226.8%) | 0.2276 | (+187.7%) | 0.0964 | (+198.8%) | 0.0355 | (+258.3%) | 0.0177 | (+258.3%) |

than"). The only discrepancy between the two batches of experiments is given by the EMQ method, which was the 3rd best in the SVM experiments while it is the worst in the L2-LR experiments; the observation made in Section 5.2 concerning the relationship between PCC and EMQ is thus pertinent here too. As a side observation, we note that in terms of absolute performance the SVM learner delivers better performance than the L2-LR learner; the difference is fairly small for all methods in {PCC, ACC, PACC, CC}, while it is very large for EMQ.

Tables 8 to 10 report the results of the L2-LR-based quantification methods on each individual dataset. If we look at these results, the superiority of PCC is even more marked than it was with the SVM-based learners: out of 66 combinations of 11 datasets $\times$ 6 evaluation measures, PCC is the best performer in 48 (2 of them tied with CC), while ACC is in 4, PACC is in 14, and CC is in 2 (tied with PCC). If we look at the average results on low-drift, mid-drift and high-drift datasets, PCC is always the best performer.

### 5.4 Statistical significance tests

In order to check whether the differences in performance among the eight quantification methods we test are statistically significant, we have performed the pairwise two-tailed Wilcoxon signed-ranks test [19, 62]) on the 11 datasets, based on the KLD results. As a non-parametric alternative to the paired t-test, the Wilcoxon test ranks the absolute differences in the performance measure (KLD, in our case) of two algorithms for each dataset, and compares the ranks for the positive and the negative differences [19]. When the computed $p$-value is lower than the significance level $\alpha = 0.05$, we reject the null hypothesis and claim that the difference of the two algorithms is statistically significant.

The results of our Wilcoxon test are displayed in Table 4. It can be observed that the reported differences between the algorithms essentially confirm, from a statistical point of view, the results in Tables 2 and 3. The superiority of PCC over most other algorithms is statistically significant, except for ACC and PACC. ACC is obtained from CC by correcting for CC's bias, i.e., for its tendency to overpredict some of the classes and underpredict the others. Such correction is very effective for CC, whose performance is rather low; this can be seen from Table 4, where ACC significantly outperforms CC. However,

**Table 4** Results of Wilcoxon tests for our 8 quantification methods, as deriving from the KLD results across the 11 datasets. Symbol "$>$" indicates that the method on the row is better, in a statistically significant way, than the method on the column; symbol "$<$" means the opposite; symbol "$\approx$" means that there is no statistically significant difference among the two; symbol "$-$" indicates that the comparison was not performed (because one of the two methods, or both, cannot be L2-LR-based). In each cell, the leftmost symbol indicates a comparison between the two SVM-based instantiations of the method, while the rightmost symbol indicates a comparison between the L2-LR-based instantiations.

| | CC | PCC | ACC | PACC | EMQ | SVM(KLD) | SVM(NKLD) | SVM(Q) |
|---|---|---|---|---|---|---|---|---|
| CC | | $<<$ | $<\approx$ | $\approx\approx$ | $<>$ | $\approx -$ | $> -$ | $> -$ |
| PCC | $>>$ | | $\approx\approx$ | $\approx\approx$ | $>>$ | $> -$ | $> -$ | $> -$ |
| ACC | $>\approx$ | $\approx\approx$ | | $\approx\approx$ | $\approx\approx$ | $> -$ | $> -$ | $> -$ |
| PACC | $\approx\approx$ | $\approx\approx$ | $\approx\approx$ | | $\approx\approx$ | $\approx -$ | $> -$ | $> -$ |
| EMQ | $><$ | $<<$ | $\approx\approx$ | $\approx\approx$ | | $\approx -$ | $> -$ | $> -$ |
| SVM(KLD) | $\approx -$ | $< -$ | $< -$ | $\approx -$ | $\approx -$ | | $> -$ | $> -$ |
| SVM(NKLD) | $< -$ | $< -$ | $< -$ | $< -$ | $< -$ | $< -$ | | $\approx -$ |
| SVM(Q) | $< -$ | $< -$ | $< -$ | $< -$ | $< -$ | $< -$ | $\approx -$ | |

from the same Table we can see that PACC is not significantly better than PCC. Similarly to ACC, PACC is obtained from PCC by correcting for PCC's bias, and the results in Table 4 indicate that PACC does not benefit from the very same correction for bias that ACC benefits from.

Anyway, the most important takeaway message to be obtained from Table 4 is that CC, the naive classification-based method that "anybody would use" when trying to estimate class prevalences, is inferior, in a statistically significant sense, to a method (PCC) explicitly designed for quantification. PCC estimates prevalences directly, without using the classification of individual items as an intermediate step (only posterior probabilities are estimated by the classifiers, and no thresholding is applied on these probabilities to obtain class assignments). This is yet another confirmation of Vapnik's principle (mentioned in Section 1), and a confirmation of the basic thesis of this paper, i.e., that when prevalence estimation is the real goal of a tweet sentiment analysis task, algorithms optimized for prevalence estimation, rather than for classification, should be employed.

## 6 Conclusion

In this paper we have argued that the real goal of most research efforts dealing with the sentiment conveyed by tweets, is not classification but quantification

(i.e., prevalence estimation). As a result, those who pursue this goal by using the learning algorithms and evaluation measures that are standard in the classification arena may obtain inaccurate prevalence estimates. We have experimentally shown, on a multiplicity of tweet sentiment classification (TSC) datasets, that more accurate prevalence estimates may be obtained by considering quantification as a task in its own right, i.e., by using quantification-specific learning algorithms which directly attempt to solve the prevalence estimation problem, rather than viewing quantification as a byproduct of classification.. Adopting a quantification-specific approach in gauging tweet sentiment may benefit many applications, especially in fields (such as political science, social science, and market research) that are usually less interested in finding the needle in the haystack than in characterising the haystack itself.

Finally, we want to note that, while this paper has addressed *sentiment* classification, the same arguments we have made apply to many studies where tweets are classified along dimensions other than sentiment. For instance, aggregate (rather than individual) results from tweet classification are the real goal in [2], which analyses Twitter data in order to predict box office revenues for movies; in [50], whose authors try to determine the percentage of tweets that are about infrastructure damage vs. those which are about donations, in order to do rapid damage assessment during humanitarian crises; in [58], where hay fever maps are generated from geo-located tweets of fever-stricken people; in [30], where the authors generate a heat map of a natural disaster from geo-located tweets that report on it; and in many others.

The present paper thus urges researchers involved in tweet mining to take the distinction between classification and prevalence estimation at heart, and optimize their systems accordingly.

## Acknowledgements

We are grateful to Chih-Chung Chang and Chih-Jen Lin for making LIBSVM available, to Rong-En Fan and colleagues for making LIBLINEAR available, to Thorsten Joachims for making SVM-perf available, to Andrea Esuli for making available the code for obtaining SVM(KLD) from SVM-perf, to José Barranquero for making available the code for obtaining SVM(Q) from SVM-perf, to Shuai Li for pointing out a small mistake in a previous version, and to Carlos Castillo for several pointers to the literature.

## References

1. Rocío Alaíz-Rodríguez, Alicia Guerrero-Curieses, and Jesús Cid-Sueiro. Class and subclass probability re-estimation to adapt a classifier in the presence of concept drift. *Neurocomputing*, 74(16):2614–2623, 2011.
2. Sitaram Asur and Bernardo A. Huberman. Predicting the future with social media. In *Proceedings of the 10th IEEE/WIC/ACM International Conference on Web Intelligence (WI 2010)*, pages 492–499, Toronto, CA, 2010.

**Table 5** Quantification accuracy (last six columns) obtained with a classification-oriented learning algorithm (CC) and seven quantification-oriented learning algorithms (PCC, ACC, PACC, EMQ, SVM(KLD), SVM(NKLD), SVM(Q)), all with SVMs as a base learner, on four low-drift STC datasets; the two bottom rows indicate the average performance of each learner across all the datasets. The Pos, Neg, Neu columns indicate (true or predicted) prevalences. **Boldface** indicates the best system.

| Dataset | System | Pos | Neu | Neg | AE | NAE | RAE | NRAE | KLD | NKLD |
|---|---|---|---|---|---|---|---|---|---|---|
| Sanders | [Training] | 0.148 | 0.691 | 0.161 | — | — | — | — | — | — |
| | [Test] | 0.148 | 0.688 | 0.164 | — | — | — | — | — | — |
| | CC | 0.080 | 0.784 | 0.137 | 0.0642 | 0.1131 | 0.2568 | 0.0998 | 0.0326 | 0.0163 |
| | PCC | 0.148 | 0.678 | 0.174 | **0.0068** | **0.0120** | **0.0270** | **0.0105** | **0.0004** | **0.0002** |
| | ACC | 0.135 | 0.677 | 0.189 | 0.0165 | 0.0291 | 0.0871 | 0.0338 | 0.0025 | 0.0012 |
| | PACC | 0.144 | 0.661 | 0.195 | 0.0209 | 0.0369 | 0.0874 | 0.0340 | 0.0033 | 0.0016 |
| | EMQ | 0.204 | 0.638 | 0.158 | 0.0369 | 0.0650 | 0.1602 | 0.0622 | 0.0103 | 0.0051 |
| | SVM(KLD) | 0.142 | 0.674 | 0.184 | 0.0134 | 0.0236 | 0.0630 | 0.0245 | 0.0014 | 0.0007 |
| | SVM(NKLD) | 0.127 | 0.566 | 0.306 | 0.0950 | 0.1674 | 0.3965 | 0.1540 | 0.0538 | 0.0269 |
| | SVM(Q) | 0.251 | 0.523 | 0.227 | 0.1098 | 0.1935 | 0.4360 | 0.1693 | 0.0575 | 0.0287 |
| WB | [Training] | 0.341 | 0.389 | 0.270 | — | — | — | — | — | — |
| | [Test] | 0.337 | 0.392 | 0.271 | — | — | — | — | — | — |
| | CC | 0.348 | 0.399 | 0.253 | 0.0135 | 0.0279 | 0.0439 | 0.0283 | 0.0011 | 0.0006 |
| | PCC | 0.335 | 0.405 | 0.260 | **0.0091** | **0.0188** | **0.0283** | **0.0182** | **0.0006** | **0.0003** |
| | ACC | 0.326 | 0.430 | 0.245 | 0.0254 | 0.0525 | 0.0768 | 0.0495 | 0.0035 | 0.0018 |
| | PACC | 0.347 | 0.436 | 0.217 | 0.0380 | 0.0784 | 0.1197 | 0.0772 | 0.0095 | 0.0047 |
| | EMQ | 0.335 | 0.417 | 0.248 | 0.0175 | 0.0361 | 0.0543 | 0.0350 | 0.0022 | 0.0011 |
| | SVM(KLD) | 0.385 | 0.300 | 0.316 | 0.0613 | 0.1266 | 0.1791 | 0.1155 | 0.0192 | 0.0096 |
| | SVM(NKLD) | 0.415 | 0.387 | 0.198 | 0.0534 | 0.1103 | 0.1756 | 0.1132 | 0.0211 | 0.0106 |
| | SVM(Q) | 0.334 | 0.355 | 0.311 | 0.0249 | 0.0515 | 0.0774 | 0.0499 | 0.0042 | 0.0021 |
| SemEval2016 | [Training] | 0.492 | 0.351 | 0.157 | — | — | — | — | — | — |
| | [Test] | 0.497 | 0.341 | 0.163 | — | — | — | — | — | — |
| | CC | 0.542 | 0.351 | 0.107 | 0.0373 | 0.0668 | 0.1555 | 0.0652 | 0.0147 | 0.0074 |
| | PCC | 0.469 | 0.360 | 0.171 | 0.0186 | 0.0333 | 0.0548 | 0.0230 | 0.0016 | 0.0008 |
| | ACC | 0.480 | 0.361 | 0.159 | **0.0136** | **0.0243** | **0.0381** | **0.0160** | **0.0009** | **0.0005** |
| | PACC | 0.501 | 0.288 | 0.211 | 0.0351 | 0.0628 | 0.1539 | 0.0646 | 0.0106 | 0.0053 |
| | EMQ | 0.452 | 0.355 | 0.193 | 0.0299 | 0.0535 | 0.1064 | 0.0446 | 0.0050 | 0.0025 |
| | SVM(KLD) | 0.439 | 0.346 | 0.215 | 0.0385 | 0.0689 | 0.1512 | 0.0635 | 0.0106 | 0.0053 |
| | SVM(NKLD) | 0.372 | 0.353 | 0.274 | 0.0830 | 0.1487 | 0.3249 | 0.1364 | 0.0457 | 0.0228 |
| | SVM(Q) | 0.400 | 0.257 | 0.343 | 0.1201 | 0.2151 | 0.5156 | 0.2164 | 0.0819 | 0.0409 |
| GASP | [Training] | 0.082 | 0.496 | 0.422 | — | — | — | — | — | — |
| | [Test] | 0.086 | 0.507 | 0.407 | — | — | — | — | — | — |
| | CC | 0.066 | 0.519 | 0.415 | 0.0138 | 0.0227 | 0.0948 | 0.0226 | 0.0032 | 0.0032 |
| | PCC | 0.087 | 0.501 | 0.412 | 0.0033 | 0.0055 | **0.0092** | **0.0022** | **0.0001** | **0.0000** |
| | ACC | 0.088 | 0.494 | 0.418 | 0.0081 | 0.0133 | 0.0226 | 0.0054 | 0.0003 | 0.0001 |
| | PACC | 0.082 | 0.508 | 0.410 | **0.0029** | **0.0048** | 0.0202 | 0.0048 | 0.0001 | 0.0001 |
| | EMQ | 0.090 | 0.510 | 0.400 | 0.0050 | 0.0082 | 0.0239 | 0.0057 | 0.0002 | 0.0001 |
| | SVM(KLD) | 0.091 | 0.481 | 0.428 | 0.0171 | 0.0281 | 0.0529 | 0.0126 | 0.0013 | 0.0007 |
| | SVM(NKLD) | 0.162 | 0.438 | 0.400 | 0.0503 | 0.0825 | 0.3416 | 0.0815 | 0.0264 | 0.0132 |
| | SVM(Q) | 0.182 | 0.452 | 0.365 | 0.0640 | 0.1051 | 0.4402 | 0.1050 | 0.0369 | 0.0184 |
| Average LD | CC | — | — | — | 0.0322 | 0.0576 | 0.1377 | 0.0540 | 0.0129 | 0.0065 |
| | PCC | — | — | — | **0.0095** | **0.0174** | **0.0298** | **0.0135** | **0.0006** | **0.0003** |
| | ACC | — | — | — | 0.0159 | 0.0298 | 0.0561 | 0.0262 | 0.0018 | 0.0009 |
| | PACC | — | — | — | 0.0242 | 0.0457 | 0.0953 | 0.0451 | 0.0059 | 0.0029 |
| | EMQ | — | — | — | 0.0223 | 0.0407 | 0.0862 | 0.0369 | 0.0044 | 0.0022 |
| | SVM(KLD) | — | — | — | 0.0326 | 0.0618 | 0.1116 | 0.0540 | 0.0081 | 0.0041 |
| | SVM(NKLD) | — | — | — | 0.0704 | 0.1272 | 0.3096 | 0.1213 | 0.0368 | 0.0184 |
| | SVM(Q) | — | — | — | 0.0797 | 0.1413 | 0.3673 | 0.1352 | 0.0451 | 0.0226 |

**Table 6** Same as Table 5, but obtained on three medium-drift datasets.

| Dataset | System | Pos | Neu | Neg | AE | NAE | RAE | NRAE | KLD | NKLD |
|---|---|---|---|---|---|---|---|---|---|---|
| OMD | [Training] | 0.266 | 0.271 | 0.463 | — | — | — | — | — | — |
| | [Test] | 0.280 | 0.283 | 0.437 | — | — | — | — | — | — |
| | CC | 0.238 | 0.225 | 0.537 | 0.0668 | 0.1391 | 0.1949 | 0.1278 | 0.0205 | 0.0102 |
| | PCC | 0.265 | 0.282 | 0.453 | **0.0107** | **0.0222** | **0.0312** | **0.0205** | **0.0007** | **0.0003** |
| | ACC | 0.268 | 0.311 | 0.420 | 0.0184 | 0.0384 | 0.0585 | 0.0383 | 0.0018 | 0.0009 |
| | PACC | 0.286 | 0.262 | 0.452 | 0.0142 | 0.0295 | 0.0440 | 0.0288 | 0.0012 | 0.0006 |
| | EMQ | 0.243 | 0.381 | 0.376 | 0.0649 | 0.1352 | 0.2047 | 0.1342 | 0.0210 | 0.0105 |
| | SVM(KLD) | 0.306 | 0.238 | 0.456 | 0.0305 | 0.0635 | 0.0999 | 0.0655 | 0.0057 | 0.0029 |
| | SVM(NKLD) | 0.331 | 0.298 | 0.371 | 0.0437 | 0.0910 | 0.1279 | 0.0839 | 0.0100 | 0.0050 |
| | SVM(Q) | 0.356 | 0.301 | 0.343 | 0.0624 | 0.1299 | 0.1826 | 0.1197 | 0.0211 | 0.0105 |
| WA | [Training] | 0.281 | 0.414 | 0.305 | — | — | — | — | — | — |
| | [Test] | 0.272 | 0.446 | 0.282 | — | — | — | — | — | — |
| | CC | 0.284 | 0.423 | 0.293 | 0.0149 | 0.0308 | 0.0437 | 0.0281 | 0.0010 | 0.0005 |
| | PCC | 0.290 | 0.405 | 0.305 | 0.0267 | 0.0551 | 0.0781 | 0.0502 | 0.0033 | 0.0017 |
| | ACC | 0.276 | 0.448 | 0.276 | **0.0039** | **0.0081** | **0.0129** | **0.0083** | **0.0001** | **0.0000** |
| | PACC | 0.280 | 0.446 | 0.273 | 0.0059 | 0.0121 | 0.0206 | 0.0132 | 0.0002 | 0.0001 |
| | EMQ | 0.297 | 0.397 | 0.306 | 0.0324 | 0.0669 | 0.0949 | 0.0610 | 0.0049 | 0.0024 |
| | SVM(KLD) | 0.175 | 0.533 | 0.292 | 0.0647 | 0.1335 | 0.1957 | 0.1258 | 0.0307 | 0.0153 |
| | SVM(NKLD) | 0.226 | 0.433 | 0.341 | 0.0393 | 0.0811 | 0.1357 | 0.0872 | 0.0100 | 0.0050 |
| | SVM(Q) | 0.327 | 0.326 | 0.348 | 0.0798 | 0.1645 | 0.2332 | 0.1498 | 0.0311 | 0.0155 |
| SemEval2013 | [Training] | 0.371 | 0.470 | 0.159 | — | — | — | — | — | — |
| | [Test] | 0.412 | 0.430 | 0.158 | — | — | — | — | — | — |
| | CC | 0.318 | 0.575 | 0.107 | 0.0965 | 0.1718 | 0.2952 | 0.1206 | 0.0431 | 0.0216 |
| | PCC | 0.360 | 0.480 | 0.160 | **0.0349** | **0.0621** | **0.0861** | **0.0352** | **0.0063** | **0.0032** |
| | ACC | 0.338 | 0.556 | 0.107 | 0.0839 | 0.1493 | 0.2660 | 0.1087 | 0.0338 | 0.0169 |
| | PACC | 0.348 | 0.544 | 0.108 | 0.0759 | 0.1351 | 0.2451 | 0.1001 | 0.0284 | 0.0142 |
| | EMQ | 0.339 | 0.543 | 0.119 | 0.0750 | 0.1335 | 0.2287 | 0.0935 | 0.0258 | 0.0129 |
| | SVM(KLD) | 0.304 | 0.538 | 0.158 | 0.0722 | 0.1286 | 0.1720 | 0.0703 | 0.0290 | 0.0145 |
| | SVM(NKLD) | 0.305 | 0.458 | 0.237 | 0.0714 | 0.1271 | 0.2756 | 0.1126 | 0.0328 | 0.0164 |
| | SVM(Q) | 0.295 | 0.479 | 0.226 | 0.0782 | 0.1393 | 0.2775 | 0.1134 | 0.0349 | 0.0175 |
| Average MD | CC | — | — | — | 0.0594 | 0.1139 | 0.1780 | 0.0922 | 0.0215 | 0.0108 |
| | PCC | — | — | — | **0.0241** | **0.0465** | **0.0651** | **0.0353** | **0.0034** | **0.0017** |
| | ACC | — | — | — | 0.0354 | 0.0653 | 0.1125 | 0.0518 | 0.0119 | 0.0060 |
| | PACC | — | — | — | 0.0320 | 0.0589 | 0.1032 | 0.0474 | 0.0099 | 0.0050 |
| | EMQ | — | — | — | 0.0574 | 0.1118 | 0.1761 | 0.0962 | 0.0172 | 0.0086 |
| | SVM(KLD) | — | — | — | 0.0558 | 0.1085 | 0.1559 | 0.0872 | 0.0218 | 0.0109 |
| | SVM(NKLD) | — | — | — | 0.0515 | 0.0997 | 0.1797 | 0.0946 | 0.0176 | 0.0088 |
| | SVM(Q) | — | — | — | 0.0735 | 0.1445 | 0.2311 | 0.1276 | 0.0290 | 0.0145 |


3. Georgios Balikas, Ioannis Partalas, Eric Gaussier, Rohit Babbar, and Massih-Reza Amini. Efficient model selection for regularized classification by exploiting unlabeled data. In *Proceedings of the 14th International Symposium on Intelligent Data Analysis (IDA 2015)*, pages 25–36, Saint Etienne, FR, 2015.
4. José Barranquero, Jorge Díez, and Juan José del Coz. Quantification-oriented learning based on reliable classifiers. *Pattern Recognition*, 48(2):591–604, 2015.
5. José Barranquero, Pablo González, Jorge Díez, and Juan José del Coz. On the study of nearest neighbor algorithms for prevalence estimation in binary problems. *Pattern Recognition*, 46(2):472–482, 2013.
6. Oscar Beijbom, Judy Hoffman, Evan Yao, Trevor Darrell, Alberto Rodriguez-Ramirez, Manuel Gonzalez-Rivero, and Ove Hoegh-Guldberg. Quantification in-the-wild: Datasets and baselines. CoRR abs/1510.04811 (2015). Presented at the NIPS 2015 Workshop on Transfer and Multi-Task Learning, Montreal, CA, 2015.
7. Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José 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.

**Table 7** Same as Table 5, but obtained on four high-drift datasets.

| Dataset | System | Pos | Neu | Neg | AE | NAE | RAE | NRAE | KLD | NKLD |
|---|---|---|---|---|---|---|---|---|---|---|
| SST | [Training] | 0.288 | 0.452 | 0.260 | — | — | — | — | — | — |
| | [Test] | 0.312 | 0.481 | 0.207 | — | — | — | — | — | — |
| | CC | 0.223 | 0.560 | 0.217 | 0.0597 | 0.1130 | 0.1671 | 0.0860 | 0.0221 | 0.0111 |
| | PCC | 0.287 | 0.465 | 0.249 | 0.0277 | 0.0523 | 0.1051 | 0.0541 | 0.0051 | 0.0025 |
| | ACC | 0.306 | 0.469 | 0.225 | **0.0121** | **0.0228** | **0.0442** | **0.0228** | **0.0010** | **0.0005** |
| | PACC | 0.305 | 0.462 | 0.232 | 0.0169 | 0.0320 | 0.0611 | 0.0314 | 0.0019 | 0.0009 |
| | EMQ | 0.278 | 0.488 | 0.234 | 0.0227 | 0.0430 | 0.0844 | 0.0434 | 0.0036 | 0.0018 |
| | SVM(KLD) | 0.304 | 0.427 | 0.269 | 0.0413 | 0.0780 | 0.1458 | 0.0750 | 0.0110 | 0.0055 |
| | SVM(NKLD) | 0.329 | 0.368 | 0.303 | 0.0749 | 0.1417 | 0.2497 | 0.1285 | 0.0331 | 0.0165 |
| | SVM(Q) | 0.306 | 0.386 | 0.308 | 0.0671 | 0.1269 | 0.2343 | 0.1206 | 0.0293 | 0.0147 |
| SemEval2015 | [Training] | 0.371 | 0.470 | 0.159 | — | — | — | — | — | — |
| | [Test] | 0.434 | 0.413 | 0.153 | — | — | — | — | — | — |
| | CC | 0.270 | 0.591 | 0.139 | 0.1188 | 0.2103 | 0.3006 | 0.1196 | 0.0726 | 0.0363 |
| | PCC | 0.321 | 0.468 | 0.211 | **0.0754** | **0.1336** | **0.2578** | **0.1025** | **0.0301** | **0.0151** |
| | ACC | 0.263 | 0.558 | 0.179 | 0.1143 | 0.2024 | 0.3060 | 0.1217 | 0.0694 | 0.0347 |
| | PACC | 0.269 | 0.539 | 0.192 | 0.1099 | 0.1946 | 0.3136 | 0.1247 | 0.0627 | 0.0314 |
| | EMQ | 0.264 | 0.541 | 0.196 | 0.1138 | 0.2015 | 0.3278 | 0.1304 | 0.0677 | 0.0338 |
| | SVM(KLD) | 0.256 | 0.519 | 0.225 | 0.1185 | 0.2099 | 0.3789 | 0.1507 | 0.0755 | 0.0377 |
| | SVM(NKLD) | 0.261 | 0.441 | 0.298 | 0.1155 | 0.2045 | 0.4720 | 0.1877 | 0.0918 | 0.0458 |
| | SVM(Q) | 0.245 | 0.473 | 0.282 | 0.1263 | 0.2236 | 0.4762 | 0.1894 | 0.0991 | 0.0495 |
| HCR | [Training] | 0.243 | 0.211 | 0.546 | — | — | — | — | — | — |
| | [Test] | 0.193 | 0.167 | 0.640 | — | — | — | — | — | — |
| | CC | 0.139 | 0.173 | 0.687 | 0.0359 | 0.0646 | 0.1300 | 0.0558 | 0.0110 | 0.0055 |
| | PCC | 0.225 | 0.205 | 0.570 | 0.0467 | 0.0841 | 0.1676 | 0.0719 | 0.0103 | 0.0051 |
| | ACC | 0.180 | 0.198 | 0.622 | 0.0208 | 0.0375 | 0.0948 | 0.0407 | 0.0033 | 0.0017 |
| | PACC | 0.171 | 0.179 | 0.650 | **0.0151** | **0.0271** | **0.0684** | **0.0293** | **0.0019** | **0.0010** |
| | EMQ | 0.163 | 0.197 | 0.640 | 0.0204 | 0.0368 | 0.1131 | 0.0485 | 0.0051 | 0.0026 |
| | SVM(KLD) | 0.131 | 0.221 | 0.648 | 0.0414 | 0.0745 | 0.2191 | 0.0941 | 0.0201 | 0.0100 |
| | SVM(NKLD) | 0.190 | 0.258 | 0.553 | 0.0604 | 0.1087 | 0.2324 | 0.0998 | 0.0248 | 0.0124 |
| | SVM(Q) | 0.269 | 0.283 | 0.449 | 0.1272 | 0.2291 | 0.4600 | 0.1975 | 0.0753 | 0.0376 |
| SemEval2014 | [Training] | 0.371 | 0.470 | 0.159 | — | — | — | — | — | — |
| | [Test] | 0.530 | 0.361 | 0.109 | — | — | — | — | — | — |
| | CC | 0.404 | 0.513 | 0.084 | 0.1010 | 0.1701 | 0.2967 | 0.0876 | 0.0464 | 0.0232 |
| | PCC | 0.439 | 0.425 | 0.136 | 0.0607 | 0.1022 | 0.1988 | 0.0587 | 0.0168 | 0.0084 |
| | ACC | 0.488 | 0.445 | 0.067 | 0.0562 | 0.0946 | 0.2341 | 0.0691 | 0.0217 | 0.0108 |
| | PACC | 0.506 | 0.454 | 0.040 | 0.0621 | 0.1045 | 0.3131 | 0.0925 | 0.0519 | 0.0259 |
| | EMQ | 0.498 | 0.430 | 0.072 | **0.0458** | **0.0772** | **0.1957** | **0.0578** | **0.0149** | **0.0074** |
| | SVM(KLD) | 0.403 | 0.473 | 0.124 | 0.0843 | 0.1419 | 0.2268 | 0.0670 | 0.0333 | 0.0167 |
| | SVM(NKLD) | 0.404 | 0.420 | 0.176 | 0.0836 | 0.1408 | 0.3367 | 0.0994 | 0.0366 | 0.0183 |
| | SVM(Q) | 0.377 | 0.452 | 0.171 | 0.1018 | 0.1713 | 0.3680 | 0.1087 | 0.0500 | 0.0250 |
| Average HD | CC | — | — | — | 0.0788 | 0.1395 | 0.2236 | 0.0872 | 0.0381 | 0.0190 |
| | PCC | — | — | — | 0.0526 | 0.0930 | 0.1823 | 0.0718 | **0.0156** | **0.0078** |
| | ACC | — | — | — | 0.0508 | **0.0893** | **0.1698** | **0.0636** | 0.0238 | 0.0119 |
| | PACC | — | — | — | 0.0510 | 0.0896 | 0.1890 | 0.0695 | 0.0296 | 0.0148 |
| | EMQ | — | — | — | **0.0507** | 0.0896 | 0.1803 | 0.0700 | 0.0228 | 0.0114 |
| | SVM(KLD) | — | — | — | 0.0714 | 0.1261 | 0.2427 | 0.0967 | 0.0350 | 0.0175 |
| | SVM(NKLD) | — | — | — | 0.0836 | 0.1489 | 0.3227 | 0.1289 | 0.0466 | 0.0233 |
| | SVM(Q) | — | — | — | 0.1056 | 0.1877 | 0.3846 | 0.1540 | 0.0634 | 0.0317 |


8. Giacomo Berardi, Andrea Esuli, and Fabrizio Sebastiani. Utility-theoretic ranking for semi-automated text classification. *ACM Transactions on Knowledge Discovery from Data*, 10(1):Article 6, 2015.
9. Johan Bollen, Huina Mao, and Xiao-Jun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.
10. Javier Borge-Holthoefer, Walid Magdy, Kareem Darwish, and Ingmar Weber. Content and network dynamics behind Egyptian political polarization on Twitter. In *Proceedings*

**Table 8** Quantification accuracy (last six columns) obtained with a classification-oriented learning algorithm (CC) and four quantification-oriented learning algorithms (PCC, ACC, PACC, EMQ), all with L2-LR as a base learner, on four low-drift STC datasets; the two bottom rows indicate the average performance of each learner across all the datasets. The Pos, Neg, Neu columns indicate (true or predicted) prevalences. **Boldface** indicates the best system.

| Dataset | System | Pos | Neu | Neg | AE | NAE | RAE | NRAE | KLD | NKLD |
|---|---|---|---|---|---|---|---|---|---|---|
| Sanders | [Training] | 0.148 | 0.691 | 0.161 | — | — | — | — | — | — |
| | [Test] | 0.148 | 0.688 | 0.164 | — | — | — | — | — | — |
| | CC | 0.077 | 0.788 | 0.135 | 0.0671 | 0.1182 | 0.2682 | 0.1042 | 0.0355 | 0.0178 |
| | PCC | 0.131 | 0.710 | 0.159 | **0.0150** | **0.0264** | **0.0602** | **0.0234** | **0.0016** | **0.0008** |
| | ACC | 0.163 | 0.637 | 0.200 | 0.0338 | 0.0595 | 0.1306 | 0.0507 | 0.0062 | 0.0031 |
| | PACC | 0.167 | 0.642 | 0.191 | 0.0301 | 0.0530 | 0.1173 | 0.0456 | 0.0046 | 0.0023 |
| | EMQ | 0.077 | 0.795 | 0.128 | 0.0715 | 0.1259 | 0.2849 | 0.1106 | 0.0383 | 0.0191 |
| WB | [Training] | 0.341 | 0.389 | 0.270 | — | — | — | — | — | — |
| | [Test] | 0.337 | 0.392 | 0.271 | — | — | — | — | — | — |
| | CC | 0.354 | 0.381 | 0.264 | 0.0132 | 0.0272 | 0.0399 | 0.0257 | **0.0009** | **0.0004** |
| | PCC | 0.341 | 0.403 | 0.255 | **0.0123** | **0.0254** | **0.0390** | **0.0252** | **0.0009** | **0.0004** |
| | ACC | 0.339 | 0.422 | 0.239 | 0.0230 | 0.0475 | 0.0719 | 0.0464 | 0.0035 | 0.0018 |
| | PACC | 0.338 | 0.413 | 0.249 | 0.0165 | 0.0340 | 0.0515 | 0.0332 | 0.0018 | 0.0009 |
| | EMQ | 0.347 | 0.416 | 0.237 | 0.0244 | 0.0503 | 0.0773 | 0.0499 | 0.0036 | 0.0018 |
| SemEval2016 | [Training] | 0.492 | 0.351 | 0.157 | — | — | — | — | — | — |
| | [Test] | 0.497 | 0.341 | 0.163 | — | — | — | — | — | — |
| | CC | 0.572 | 0.316 | 0.113 | 0.0500 | 0.0895 | 0.1771 | 0.0743 | 0.0158 | 0.0079 |
| | PCC | 0.441 | 0.339 | 0.220 | **0.0379** | **0.0680** | **0.1553** | **0.0652** | **0.0115** | **0.0057** |
| | ACC | 0.512 | 0.259 | 0.228 | 0.0542 | 0.0970 | 0.2246 | 0.0943 | 0.0223 | 0.0112 |
| | PACC | 0.527 | 0.211 | 0.262 | 0.0864 | 0.1547 | 0.3504 | 0.1470 | 0.0561 | 0.0280 |
| | EMQ | 0.594 | 0.293 | 0.113 | 0.0646 | 0.1157 | 0.2126 | 0.0892 | 0.0215 | 0.0107 |
| GASP | [Training] | 0.082 | 0.496 | 0.422 | — | — | — | — | — | — |
| | [Test] | 0.086 | 0.507 | 0.407 | — | — | — | — | — | — |
| | CC | 0.058 | 0.526 | 0.416 | 0.0189 | 0.0311 | 0.1297 | 0.0310 | 0.0065 | 0.0032 |
| | PCC | 0.072 | 0.508 | 0.421 | 0.0097 | 0.0160 | 0.0682 | 0.0163 | 0.0016 | 0.0008 |
| | ACC | 0.109 | 0.499 | 0.392 | 0.0150 | 0.0246 | 0.1038 | 0.0248 | 0.0028 | 0.0014 |
| | PACC | 0.073 | 0.514 | 0.413 | **0.0087** | **0.0142** | **0.0597** | **0.0142** | **0.0012** | **0.0006** |
| | EMQ | 0.052 | 0.525 | 0.424 | 0.0231 | 0.0379 | 0.1589 | 0.0379 | 0.0102 | 0.0051 |
| Average LD | CC | — | — | — | 0.0373 | 0.0665 | 0.1537 | 0.0588 | 0.0147 | 0.0073 |
| | PCC | — | — | — | **0.0187** | **0.0339** | **0.0807** | **0.0325** | **0.0039** | **0.0019** |
| | ACC | — | — | — | 0.0315 | 0.0572 | 0.1327 | 0.0540 | 0.0087 | 0.0044 |
| | PACC | — | — | — | 0.0354 | 0.0640 | 0.1447 | 0.0600 | 0.0159 | 0.0080 |
| | EMQ | — | — | — | 0.0459 | 0.0824 | 0.1834 | 0.0719 | 0.0184 | 0.0092 |

*of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW 2015)*, pages 700–711, Vancouver, CA, 2015.

11. S. Burton and A. Soboleva. Interactive or reactive? Marketing with Twitter. *Journal of Consumer Marketing*, 28(7):491–499, 2011.
12. Yee Seng Chan and Hwee Tou Ng. Estimating class priors in domain adaptation for word sense disambiguation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL 2006)*, pages 89–96, Sydney, AU, 2006.
13. Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):Article 27, 2011.
14. Bryan R. Conroy and Paul Sajda. Fast, exact model selection and permutation testing for L2-regularized logistic regression. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS 2012)*, pages 246–254, La Palma, ES, 2012.

**Table 9** Same as Table 8, but obtained on three medium-drift datasets.

| Dataset | System | Pos | Neu | Neg | AE | NAE | RAE | NRAE | KLD | NKLD |
|---|---|---|---|---|---|---|---|---|---|---|
| OMD | [Training] | 0.266 | 0.271 | 0.463 | — | — | — | — | — | — |
| | [Test] | 0.280 | 0.283 | 0.437 | — | — | — | — | — | — |
| | CC | 0.262 | 0.223 | 0.516 | 0.0524 | 0.1092 | 0.1527 | 0.1001 | 0.0146 | 0.0073 |
| | PCC | 0.276 | 0.290 | 0.434 | **0.0046** | **0.0095** | **0.0095** | **0.0099** | **0.0001** | **0.0001** |
| | ACC | 0.267 | 0.319 | 0.414 | 0.0239 | 0.0499 | 0.0753 | 0.0494 | 0.0030 | 0.0015 |
| | PACC | 0.293 | 0.285 | 0.422 | 0.0100 | 0.0208 | 0.0293 | 0.0192 | 0.0006 | 0.0003 |
| | EMQ | 0.267 | 0.199 | 0.534 | 0.0648 | 0.1349 | 0.1886 | 0.1237 | 0.0256 | 0.0128 |
| WA | [Training] | 0.281 | 0.414 | 0.305 | — | — | — | — | — | — |
| | [Test] | 0.272 | 0.446 | 0.282 | — | — | — | — | — | — |
| | CC | 0.308 | 0.380 | 0.312 | 0.0434 | 0.0894 | 0.1270 | 0.0816 | 0.0088 | 0.0044 |
| | PCC | 0.298 | 0.395 | 0.307 | 0.0338 | 0.0697 | 0.0990 | 0.0636 | 0.0053 | 0.0027 |
| | ACC | 0.318 | 0.384 | 0.298 | 0.0407 | 0.0839 | 0.1197 | 0.0769 | 0.0084 | 0.0042 |
| | PACC | 0.306 | 0.404 | 0.291 | **0.0277** | **0.0570** | **0.0815** | **0.0523** | **0.0040** | **0.0020** |
| | EMQ | 0.305 | 0.387 | 0.309 | 0.0391 | 0.0806 | 0.1145 | 0.0736 | 0.0072 | 0.0036 |
| SemEval2013 | [Training] | 0.371 | 0.470 | 0.159 | — | — | — | — | — | — |
| | [Test] | 0.412 | 0.430 | 0.158 | — | — | — | — | — | — |
| | CC | 0.319 | 0.579 | 0.102 | 0.0996 | 0.1774 | 0.3095 | 0.1265 | 0.0468 | 0.0234 |
| | PCC | 0.371 | 0.420 | 0.209 | **0.0344** | **0.0613** | **0.1506** | **0.0616** | **0.0093** | **0.0047** |
| | ACC | 0.335 | 0.551 | 0.114 | 0.0806 | 0.1435 | 0.2479 | 0.1013 | 0.0300 | 0.0150 |
| | PACC | 0.344 | 0.552 | 0.104 | 0.0812 | 0.1446 | 0.2626 | 0.1073 | 0.0327 | 0.0164 |
| | EMQ | 0.318 | 0.609 | 0.074 | 0.1191 | 0.2121 | 0.3923 | 0.1603 | 0.0780 | 0.0390 |
| Average MD | CC | — | — | — | 0.0651 | 0.1253 | 0.1964 | 0.1027 | 0.0234 | 0.0117 |
| | PCC | — | — | — | **0.0243** | **0.0469** | **0.0864** | **0.0450** | **0.0049** | **0.0025** |
| | ACC | — | — | — | 0.0484 | 0.0924 | 0.1476 | 0.0759 | 0.0138 | 0.0069 |
| | PACC | — | — | — | 0.0396 | 0.0741 | 0.1244 | 0.0596 | 0.0124 | 0.0062 |
| | EMQ | — | — | — | 0.0743 | 0.1425 | 0.2318 | 0.1192 | 0.0369 | 0.0184 |


15. Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. John Wiley & Sons, New York, US, 1991.
16. Imre Csiszár and Paul C. Shields. Information theory and statistics: A tutorial. *Foundations and Trends in Communications and Information Theory*, 1(4):417–528, 2004.
17. Giovanni Da San Martino, Wei Gao, and Fabrizio Sebastiani. QCRI at SemEval-2016 Task 4: Probabilistic methods for binary and ordinal quantification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, San Diego, US, 2016. Forthcoming.
18. Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39(1):1–38, 1977.
19. Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.
20. Peter S. Dodds, Kameron D. Harris, Isabel M. Kloumann, Catherine A. Bliss, and Christopher M. Danforth. Temporal patterns of happiness and information in a global social network: Hedonometrics and Twitter. *PLoS ONE*, 6(12), 2011.
21. Andrea Esuli. ISTI-CNR at SemEval-2016 Task 4: Quantification on an ordinal scale. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, San Diego, US, 2016.
22. Andrea Esuli and Fabrizio Sebastiani. Sentiment quantification. *IEEE Intelligent Systems*, 25(4):72–75, 2010.
23. Andrea Esuli and Fabrizio Sebastiani. Explicit loss minimization in quantification applications (preliminary draft). Presented at the *8th International Workshop on Information Filtering and Retrieval (DART 2014)*, Pisa, IT, 2014.
24. Andrea Esuli and Fabrizio Sebastiani. Optimizing text quantifiers for multivariate loss functions. *ACM Transactions on Knowledge Discovery and Data*, 9(4):Article 27, 2015.

**Table 10** Same as Table 8, but obtained on four high-drift datasets.

| Dataset | System | Pos | Neu | Neg | AE | NAE | RAE | NRAE | KLD | NKLD |
|---|---|---|---|---|---|---|---|---|---|---|
| SST | [Training] | 0.288 | 0.452 | 0.260 | — | — | — | — | — | — |
| | [Test] | 0.312 | 0.481 | 0.207 | — | — | — | — | — | — |
| | CC | 0.263 | 0.490 | 0.247 | 0.0330 | 0.0624 | 0.1239 | 0.0638 | 0.0079 | 0.0040 |
| | PCC | 0.270 | 0.488 | 0.242 | **0.0282** | **0.0534** | **0.1068** | **0.0550** | **0.0059** | **0.0030** |
| | ACC | 0.315 | 0.407 | 0.279 | 0.0492 | 0.0931 | 0.1689 | 0.0869 | 0.0164 | 0.0082 |
| | PACC | 0.201 | 0.607 | 0.192 | 0.0841 | 0.1591 | 0.2302 | 0.1185 | 0.0410 | 0.0205 |
| | EMQ | 0.257 | 0.512 | 0.231 | 0.0369 | 0.0698 | 0.1190 | 0.0613 | 0.0078 | 0.0039 |
| SemEval2015 | [Training] | 0.371 | 0.470 | 0.159 | — | — | — | — | — | — |
| | [Test] | 0.434 | 0.413 | 0.153 | — | — | — | — | — | — |
| | CC | 0.289 | 0.578 | 0.133 | 0.1101 | 0.1950 | 0.2879 | 0.1145 | 0.0593 | 0.0297 |
| | PCC | 0.365 | 0.418 | 0.217 | **0.0460** | **0.0815** | **0.1973** | **0.0785** | **0.0168** | **0.0084** |
| | ACC | 0.275 | 0.539 | 0.186 | 0.1064 | 0.1885 | 0.2971 | 0.1182 | 0.0587 | 0.0293 |
| | PACC | 0.282 | 0.540 | 0.177 | 0.1013 | 0.1793 | 0.2729 | 0.1085 | 0.0532 | 0.0266 |
| | EMQ | 0.261 | 0.593 | 0.145 | 0.1204 | 0.2131 | 0.2949 | 0.1173 | 0.0787 | 0.0393 |
| HCR | [Training] | 0.243 | 0.211 | 0.546 | — | — | — | — | — | — |
| | [Test] | 0.193 | 0.167 | 0.640 | — | — | — | — | — | — |
| | CC | 0.132 | 0.149 | 0.719 | 0.0525 | 0.0946 | 0.1817 | 0.0780 | 0.0179 | 0.0090 |
| | PCC | 0.196 | 0.173 | 0.631 | **0.0055** | **0.0100** | **0.0202** | **0.0087** | **0.0002** | **0.0001** |
| | ACC | 0.183 | 0.203 | 0.614 | 0.0240 | 0.0433 | 0.1026 | 0.0440 | 0.0042 | 0.0021 |
| | PACC | 0.177 | 0.216 | 0.606 | 0.0329 | 0.0592 | 0.1436 | 0.0616 | 0.0077 | 0.0039 |
| | EMQ | 0.088 | 0.138 | 0.774 | 0.0895 | 0.1612 | 0.3093 | 0.1328 | 0.0622 | 0.0311 |
| SemEval2014 | [Training] | 0.371 | 0.470 | 0.159 | — | — | — | — | — | — |
| | [Test] | 0.530 | 0.361 | 0.109 | — | — | — | — | — | — |
| | CC | 0.406 | 0.517 | 0.077 | 0.1043 | 0.1755 | 0.3212 | 0.0949 | 0.0497 | 0.0249 |
| | PCC | 0.380 | 0.420 | 0.201 | 0.1001 | 0.1685 | 0.4277 | 0.1263 | 0.0557 | 0.0278 |
| | ACC | 0.484 | 0.448 | 0.068 | 0.0581 | 0.0978 | **0.2360** | **0.0697** | **0.0219** | **0.0110** |
| | PACC | 0.505 | 0.441 | 0.054 | **0.0533** | **0.0898** | 0.2573 | 0.0760 | 0.0298 | 0.0149 |
| | EMQ | 0.476 | 0.482 | 0.042 | 0.0807 | 0.1358 | 0.3517 | 0.1039 | 0.0573 | 0.0287 |
| Average HD | CC | — | — | — | 0.0750 | 0.1319 | 0.2287 | 0.0878 | 0.0337 | 0.0169 |
| | PCC | — | — | — | **0.0450** | **0.0783** | **0.1880** | **0.0671** | **0.0197** | **0.0098** |
| | ACC | — | — | — | 0.0594 | 0.1057 | 0.2012 | 0.0797 | 0.0253 | 0.0127 |
| | PACC | — | — | — | 0.0679 | 0.1219 | 0.2260 | 0.0912 | 0.0329 | 0.0165 |
| | EMQ | — | — | — | 0.0819 | 0.1450 | 0.2687 | 0.1038 | 0.0515 | 0.0257 |


25. Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.
26. George 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.
27. George Forman. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, 17(2):164–206, 2008.
28. Wei Gao and Fabrizio Sebastiani. Tweet sentiment: From classification to quantification. In *Proceedings of the 7th International Conference on Advances in Social Network Analysis and Mining (ASONAM 2015)*, pages 97–104, Paris, FR, 2015.
29. Víctor González-Castro, Rocío Alaiz-Rodríguez, and Enrique Alegre. Class distribution estimation based on the Hellinger distance. *Information Sciences*, 218:146–164, 2013.
30. Benjamin Herfort, Svend-Jonas Schelhorn, João P. de Albuquerque, and Alexander Zipf. Does the spatiotemporal distribution of tweets match the spatiotemporal distribution of flood phenomena? A study about the river Elbe flood in June 2013. In *Proceedings of the 11th International Conference on Information Systems for Crisis Response and Management (ISCRAM 2014)*, pages 747–751, Philadelphia, US, 2014.
31. Daniel J. Hopkins and Gary King. A method of automated nonparametric content analysis for social science. *American Journal of Political Science*, 54(1):229–247, 2010.

32. Thorsten 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.
33. Thorsten Joachims, Thomas Hofmann, Yisong Yue, and Chun-Nam Yu. Predicting structured objects with support vector machines. *Communications of the ACM*, 52(11):97–104, 2009.
34. Mesut Kaya, Guven Fidan, and Ismail Hakki Toroslu. Transfer learning using Twitter data for improving sentiment classification of Turkish political news. In *Proceedings of the 28th International Symposium on Computer and Information Sciences (ISCIS 2013)*, pages 139–148, Paris, FR, 2013.
35. Gary King and Ying Lu. Verbal autopsy methods with multiple causes of death. *Statistical Science*, 23(1):78–91, 2008.
36. Svetlana Kiritchenko, Xiaodan Zhu, and Saif M. Mohammad. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762, 2014.
37. Patrice Latinne, Marco Saerens, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities may significantly improve classification accuracy: Evidence from a multi-class problem in remote sensing. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, pages 298—305, Williamstown, US, 2001.
38. David D. Lewis. Evaluating and optimizing 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.
39. Nachai Limsetto and Kitsana Waiyamai. Handling concept drift via ensemble and class distribution estimation technique. In *Proceedings of the 7th International Conference on Advanced Data Mining (ADMA 2011)*, pages 13–26, Bejing, CN, 2011.
40. Micol Marchetti-Bowick and Nathanael Chambers. Learning for microblogs with distant supervision: Political forecasting with Twitter. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)*, pages 603–612, Avignon, FR, 2012.
41. Eugenio Martínez-Cámara, Maria Teresa Martín-Valdivia, Luis Alfonso Ureña López, and Arturo Montejo Ráez. Sentiment analysis in Twitter. *Natural Language Engineering*, 20(1):1–28, 2014.
42. Yelena Mejova, Ingmar Weber, and Michael W. Macy, editors. *Twitter: A Digital Socioscope*. Cambridge University Press, Cambridge, UK, 2015.
43. Letizia Milli, Anna Monreale, Giulio Rossetti, Fosca Giannotti, Dino Pedreschi, and Fabrizio Sebastiani. Quantification trees. In *Proceedings of the 13th IEEE International Conference on Data Mining (ICDM 2013)*, pages 528–536, Dallas, US, 2013.
44. Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, pages 321–327, Atlanta, US, 2013.
45. Kevin P. Murphy. *Machine learning. A probabilistic perspective*. The MIT Press, Cambridge, US, 2012.
46. Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. SemEval-2016 Task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, San Diego, US, 2016. Forthcoming.
47. Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. SemEval-2013 Task 2: Sentiment analysis in Twitter. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, pages 312–320, Atlanta, US, 2013.
48. Harikrishna Narasimhan, Shuai Li, Purushottam Kar, Sanjay Chawla, and Fabrizio Sebastiani. Stochastic optimization techniques for quantification performance measures. 2016. Submitted for publication.
49. Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 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 2010)*, Washington, US, 2010.

50. Alexandra Olteanu, Sarah Vieweg, and Carlos Castillo. What to expect when the unexpected happens: Social media communications across crises. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW 2015)*, pages 994–1009, Vancouver, CA, 2015.
51. Weike Pan, Erheng Zhong, and Qiang Yang. Transfer learning for text mining. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 223–258. Springer, Heidelberg, DE, 2012.
52. Muhammad A. Qureshi, Colm O'Riordan, and Gabriella Pasi. Clustering with error estimation for monitoring reputation of companies on Twitter. In *Proceedings of the 9th Asia Information Retrieval Societies Conference (AIRS 2013)*, pages 170–180, Singapore, SN, 2013.
53. Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. SemEval-2015 Task 10: Sentiment analysis in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 451–463, Denver, US, 2015.
54. Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. SemEval-2014 Task 9: Sentiment analysis in Twitter. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 73–80, Dublin, IE, 2014.
55. Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14(1):21–41, 2002.
56. Hassan Saif, Miriam Fernez, Yulan He, and Harith Alani. Evaluation datasets for Twitter sentiment analysis: A survey and a new dataset, the STS-Gold. In *Proceedings of the 1st International Workshop on Emotion and Sentiment in Social and Expressive Media (ESSEM 2013)*, pages 9–21, Torino, IT, 2013.
57. Lidia Sánchez, Víctor González, Enrique Alegre, and Rocío 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.
58. Tetsuro Takahashi, Shuya Abe, and Nobuyuki Igata. Can Twitter be an alternative of real-world sensors? In *Proceedings of the 14th International Conference on Human-Computer Interaction (HCI International 2011)*, pages 240–249, Orlando, US, 2011.
59. Lei Tang, Huiji Gao, and Huan 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.
60. Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.
61. Vladimir Vapnik. *Statistical Learning Theory*. Wiley, New York, US, 1998.
62. Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.
63. Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5:975–1005, 2004.
64. Jack Chongjie Xue and Gary 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.
65. Zhihao Zhang and Jie Zhou. Transfer estimation of evolving class priors in data stream classification. *Pattern Recognition*, 43(9):3151–3161, 2010.
66. Xiaodan Zhu, Svetlana Kiritchenko, and Saif M. Mohammad. NRC-Canada-2014: Recent improvements in the sentiment analysis of tweets. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 443–447, Dublin, IE, 2014.
67. Fuhao Zou, Yunfei Wang, Yang Yang, Ke Zhou, Yunpeng Chen, and Jingkuan Song. Supervised feature learning via L2-norm regularized logistic regression for 3D object recognition. *Neurocomputing*, 151:603–611, 2015.