[redacted]

Sentiment Quantification of User-Generated Content

Fabrizio Sebastiani
Istituto di Scienza e Tecnologie dell’Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy

Synonyms

Estimating prevalence of sentiment classes in user-generated content

Glossary

Prevalence of c in set $\mathcal{S}$	Percentage of items in $\mathcal{S}$ that belong to class c and also known as the “relative frequency” of c or the “prior probability” (or simply “prior”) of c
Quantification	Estimation of the prevalence of each class $c \in \mathcal{C}$ in a set $\mathcal{S}$ of unlabeled items (or estimation of the distribution of $\mathcal{S}$ across the classes in $\mathcal{C}$), synonym of “supervised prevalence estimation” and “class prior estimation,” and also previously referred to as “counting.”
Sentiment classification	A classification task whereby items (e.g., tweets, product reviews, comments, answers to open-ended questions) are classified based on the sentiment they convey (or opinion they express) about a certain entity or topic. It may take the form of binary classification (when the available classes are $\mathcal{C} =$ {Positive, Negative}) or ternary classification (when $\mathcal{C}$ also contains a Neutral class) or ordinal classification (when there are more than two classes, and these classes are ordered according to a total order, e.g., Excellent, Good, Fair, Poor, Disastrous)

Definition

User-generated content (UGC) is defined as content (usually in the form of text, spoken audio, imagery, video, etc.) authored by casual users (as opposed to professional users) of a digital content delivery platform. Examples of user-generated textual content are tweets, blog posts, product reviews, and as are all of the comments that other users upload as a reaction to them; examples of user-generated non-textual content are images uploaded on Instagram (or similar social networking services) or videos uploaded on platforms such as YouTube. *Sentiment quantification* of UGC is defined as the activity (carried out via supervised learning) of estimating the prevalence (aka percentage or relative frequency) of sentiment-related classes (e.g., Positive, Negative, Neutral) in a set of unlabeled UGC items.

Introduction

User-generated content (UGC) has turned into a goldmine for market researchers, social scientists, political scientists, and professionals involved in reputation management, since it gives near-instant access to a potentially enormous quantity of data from which the collective sentiment about products, companies, policies, and political candidates can be gauged.

Possibly the most important task underlying attempts to tap into this goldmine is *sentiment classification*, the task of classifying an item of UGC (e.g., a tweet, a product review, a post on a social networking service) according to the sentiment it conveys (or opinion it expresses) about a certain entity or topic. From a niche, esoteric topic that only a handful of NLP researchers were investigating, in the last 10 years, sentiment analysis (and sentiment classification, which is its most prominent incarnation) has blossomed into a research field with thousands of active researchers and into a multimillion industry too (almost all providers of textual content analysis tools nowadays boast a sentiment analysis solution as part of their commercial offer).

However, it turns out that in many applications, the final goal of sentiment classification is not that of determining the class of individual UGC items but that of estimating the prevalence of UGC items that belong to a certain class; the latter is a more specific task than the former, since a solution for the former is also a solution for the latter, but not vice versa. When prevalence estimation is tackled via supervised learning, it is known as *quantification*. Quantification has recently been investigated as a task of its own (i.e., as something which is not a mere by-product of classification), following experimental evidence that using quantification-specific algorithms, rather than standard classification-oriented ones, delivers superior quantification accuracy.

We here give an introduction to the task of quantifying UGC by sentiment, to the methods that have been proposed in the literature, and to the measures that have been used for evaluating the accuracy of different quantification algorithms.

Key Points

The obvious method for dealing with quantification is "classify and count," i.e., classifying each unlabeled object via a standard classifier and estimating class prevalence by counting the objects that have been labeled 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, which it has generated on a given dataset) is worse, in terms of classification accuracy, than a classifier h_2 for which, on the same dataset, $FP = 18$ 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., Barranquero et al. 2015; Bella et al. 2010; Esuli and Sebastiani 2015; Forman 2008) 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 (since quantification can be framed in terms of classification, 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" (Vapnik 1998), 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 the rest of this work, we will thus look at approaches that have tackled quantification as a task of its own rather than as a by-product of classification.

Historical Background

Sentiment quantification for UGC is a task at the crossroads of two main streams or research, namely, (a) sentiment analysis of UGC and (b) quantification.

Sentiment analysis (see Feldman 2013; Liu 2012; Pang and Lee 2008) is a fairly recent task, since until 15 years ago, it had only resulted in sporadic efforts, mainly originating from the NLP area. It was actually the rising importance of UGC, caused by the birth of new modes of expression (e.g., blogs, user-contributed product reviews) and new platforms for hosting them (e.g., electronic commerce portals, social networking services) that contributed to the growing importance of sentiment analysis. Indeed, while the Web of the 1990s was primarily a repository of factual content generated by professional authors, the Web of the new millennium has progressively become rife with opinion-laden content generated by casual users, and it is the sentiment-laden nature of this content that has prompted the explosion of sentiment analysis.

Earlier efforts at sentiment analysis were extremely primitive, some of them consisting of algorithms that counted the number p of occurrences of "positive words" (i.e., words that conveyed a sense of positivity, such as "truthful," "exceptional," and the like) and the number n of occurrences of "negative words" (i.e., words that conveyed a sense of negativity, such as "inaccurate," "pathetic," and the like) within a textual UGC item and decreed the item a Positive one if $p > n$ and a Negative one otherwise. This phase could be preceded by a check that $p + n$ exceeded a certain threshold, failing which the item was decreed a Neutral one. These primitive efforts met with some success, but clashed against the lack of high-coverage, manually crafted *sentiment lexicons*, i.e., dictionaries where each word of a given language is classified as a positive or a negative or a neutral word. This encouraged many researchers to devote their attention to devising methods for the automatic (or semiautomatic) extraction of such lexicons (also for languages other than English) from textual data.

The last 10 years have seen sentiment analysis researchers adopt increasingly sophisticated tools for linguistic analysis and text mining, thus leaving behind the "counting positive and negative words" phase; some of these tools will be discussed in section "Representing Sentiment for User-Generated Content."

Quantification has instead a more complex history, and it is fair to say that different strands in the quantification literature evolved almost independently (and somehow unknown to each other) within (a) statistics (e.g., Hopkins and King 2010), (b) machine learning (e.g., du Plessis and Sugiyama 2012; Saerens et al. 2002), and (c) data mining (e.g., Forman 2008). One interesting fact is that in strand (b), prevalence estimation is not a goal in itself but is functional to improving the accuracy of a classifier in situations characterized by *distribution drift* (i.e., by class prevalences that are likely to change substantially from the training set to the test set); instead, in strand (c) prevalence estimation is a goal in itself (the case of UGC that we are analyzing here belongs to this latter case).

Different quantification algorithms have been proposed over the years (on this, see section "Learning to Quantify" for more details). Most of them are batch learning methods, which require all the training examples to be loaded in memory at the same time, while incremental "online" methods, which relax this requirement and thus start learning after the first training examples are loaded in memory, are a rarity (see Kar et al. 2016 for an example).

The first work where sentiment analysis meets quantification is Esuli and Sebastiani (2010b), which observes that many applications of sentiment classification really have quantification, and not classification, as their goal, and thus argues for the importance of developing learning algorithms that explicitly target quantification and not classification. See section "Key Applications" for other pointers to the literature on sentiment quantification for UGC.

Main Approaches to Sentiment Quantification

Setting up a system that performs sentiment quantification of UGC essentially involves setting up two software modules:

1. A module that generates vectorial representations of UGC items (e.g., blog posts, tweets) which can be fed to an algorithm which learns a quantifier from training data
2. A module that learns a quantifier from the vectorial representations of labeled UGC items.

After discussing in section "Evaluating Quantification" the main measures that are used in the literature for evaluating quantification, in sections "Representing Sentiment for User-Generated Content" and "Learning to Quantify," we discuss the main techniques for building modules 1 and 2, respectively.

Evaluating Quantification

Different measures have been proposed in the literature for evaluating quantification error. We here concentrate on the measures that have been proposed for tackling *single-label multi-class* (SLMC) quantification. 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.

Notation-wise, by $\Lambda(p, \hat{p}, \mathcal{S}, \mathcal{C})$ we will indicate a *quantification loss*, i.e., a measure Λ 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. (Consistently with most mathematical literature, we use the caret symbol (ˆ) to indicate estimation.)

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

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

(worst). The main advantage of AE is that it is intuitive and easy to understand to non-initiates too.

However, AE does 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 relativizing the value $|\hat{p}(c) - p(c)|$ in Eq. 1 to the true class prevalence, i.e.,

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

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.,

$$p_s(c) = \frac{\varepsilon + p(c)}{\varepsilon|\mathcal{C}| + \sum_{c\in\mathcal{C}} p(c)} \tag{3}$$

where $p_s(c)$ denotes the smoothed version of $p(c)$, the denominator is just a normalizing factor, and where the quantity $\varepsilon = \frac{1}{2|\mathcal{S}|}$ is typically used as a smoothing factor (Esuli and Sebastiani 2015; Forman 2008; Gao and Sebastiani 2015); $\hat{p}_s(c)$, the smoothed version of the predicted class prevalence, is defined analogously. The smoothed versions of $p(c)$ and $\hat{p}(c)$ are then used in place of their original versions in Eq. 2; 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

$$\frac{|\mathcal{C}| - 1 + \dfrac{1 - \min\limits_{c \in \mathcal{C}} p_s(c)}{\min\limits_{c \in \mathcal{C}} p_s(c)}}{|\mathcal{C}|}$$

(worst).

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., Cover and Thomas 1991). KLD was proposed as a SLMC quantification measure in Forman (2005) and is defined as

$$\mathrm{KLD}(p,\hat{p}) = \sum_{c \in \mathcal{C}} p(c) \log_e \frac{p(c)}{\hat{p}(c)} \qquad (4)$$

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 Eq. 4 has predicted prevalences, and not true prevalences, at the denominator: that is, by making a predicted prevalence $\hat{p}(c)$ infinitely small, we can make KLD be infinitely large.

Also KLD 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, we use the smoothed prevalences of Eq. 3; as a result, KLD is always defined and still returns 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 (Csiszar and Shields 2004) and, although from a more applicative angle, within the language modeling approach to information retrieval and to speech processing. As a consequence, it has emerged as the *de facto* standard in the SLMC quantification literature.

Concerning ordinal quantification, the only known measure is the *Earth Mover's Distance* (EMD), a measure well-known in the field of computer vision. It is defined for the general case in which a distance $d(c', c'')$ is defined for each $c', c'' \in \mathcal{C}$; when there is a total order on the classes in $\mathcal{C}$, if we assume that $d(c_i, c_{i+1}) = 1$ for all $i \in \{1, \ldots, (\mathcal{C} - 1)\}$, the Earth Mover's Distance is defined as

$$\mathrm{EMD}(p,\hat{p}) = \sum_{j=1}^{|\mathcal{C}|-1} \left| \sum_{i=1}^{j} \hat{p}(c_i) - \sum_{i=1}^{j} p(c_i) \right| \qquad (5)$$

and can be computed in $|\mathcal{C}|$ steps from the estimated and true class prevalences. EMD ranges between 0 (best) and $(|\mathcal{C}| - 1)$ (worst).

Representing Sentiment for User-Generated Content

For quantification, sentiment analysis comes into play when generating vectorial representations for the labeled examples that are used to generate a sentiment quantifier and for the unlabeled examples that are the object of quantification. For generating these representations, no technique specific to sentiment *quantification* has emerged, which means that the same techniques used for sentiment *classification* are also used for quantification.

The traditional “bag of words” (BoW) approach to representing textual content in classification by topic cannot be used for classifying by sentiment: to see why, simply consider the fact that two sentences such as “A horrible hotel in a beautiful town” and “A beautiful hotel in a horrible town” would be assigned the same class if relying on a BoW representation. As a result, classification by sentiment relies on more sophisticated linguistic tools than classification by topic; these tools include:

- Part-of-speech taggers (e.g., to detect the difference between “good” as an adjective and “good” as a noun)
- Valence shifter detectors (e.g., to detect the presence of negated contexts, since negation usually inverts the polarity of the sentiment expressed in the scope of the negation)
- Detectors of amplifiers (e.g., “very”) and diminishers (e.g., “scarcely”), since they change the intensity of any sentiment expressed within their scope
- Parsers and named entity recognizers (e.g., to detect which entity or entity sentiments are about)

One linguistic tool of particular importance in sentiment analysis is a sentiment lexicon, i.e., a dictionary (or thesaurus) where lexical entries are tagged according to whether they carry positive sentiment (e.g., the adjective “phenomenal”), negative sentiment (e.g., the adjective “disappointing”), or no sentiment at all (e.g., the adjective “electronic”). The availability of a sentiment lexicon is crucially important, e.g., to distinguish sentences such as “The room had a comfortable bed” (which indeed carries positive sentiment toward the hotel being reviewed) from sentences such as “The room had a rectangular bed” (which carries no sentiment).

Sentiment analysis, like many other NLP tasks, has not been indifferent to the “deep learning revolution” that has swept the field of machine learning in the last 5 years. In sentiment analysis, the most visible effect of this revolution has been the adoption of “word embeddings,” which allow the vectorial representations of UGC items to leverage the distributional semantics of the words occurring in them (Tang et al. 2014).

An important fact to be noted is that UGC, because of its informal nature, is often fraught with syntactic inaccuracies, abbreviations, typos, slang expressions, etc.; these features make the linguistic analysis (and, as a consequence, the analysis by sentiment) of UGC more difficult than the analysis of more formal, polished language. It is thus often beneficial to use linguistic analysis tools (including sentiment lexicons) that are UGC specific and often even specific to the particular medium to be analyzed (e.g., Twitter).

Learning to Quantify

In the last 10 years several supervised learning approaches to prevalence estimation have been proposed, the two main classes being the *aggregative* and the *non-aggregative* methods. While the former requires the classification of each individual item as an intermediate step, the latter does not and estimates class prevalences holistically. Most methods fall in the former class, while the latter has few representatives (e.g., Gonzalez-Castro et al. 2013; King and Lu 2008).

We here introduce in some detail a few representative approaches of the aggregative type. Let us fix some notation. We assume a domain $\mathcal{X}$ of UGC items; a generic item will be indicated by $\mathbf{x}$. We assume the availability of a set *Tr* of training items and of a set *Te* of test items in which the accuracy of a quantifier is evaluated. A classifier (or *hypothesis*) trained on *Tr* will be denoted by $h\mathcal{X} \rightarrow \mathcal{C}$, where $\mathcal{X}$ is our domain of interest (e.g., the set of tweets) and $\mathcal{C}$ is the set of classes (e.g., Positive, Negative, Neutral). By $\widehat{p}_S^M(c)$ we denote the true prevalence of class $c \in \mathcal{C}$ in set $\mathcal{S}$, while by $\widehat{p}_S^M(c)$ we denote the prevalence of class $c \in \mathcal{C}$ in set $\mathcal{S}$ 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 items in *Te*, and estimating p_{Te} by simply counting the fraction of items in *Te* that are predicted to belong to the class. If by $\widehat{c}$ we denote the event “class c has been assigned by the classifier,” so that $p_{Te}(\widehat{c})$ represents the fraction of test documents that have been assigned to c by the classifier, this corresponds to computing

$$\hat{p}_{Te}^{CC}(c) = p_{Te}(\hat{c}) = \frac{|\{\mathbf{x} \in Te | h(\mathbf{x}) = c\}|}{|Te|} \quad (6)$$

Forman (2008) calls this the *classify and count* (CC) method. This is the classification-oriented method that often uses as a baseline in quantification experiments.

Probabilistic Classify and Count (PCC). A variant of CC consists in generating a classifier from *Tr*, classifying the items in *Te*, and computing $p_{Te}(c)$ as the *expected* fraction of items 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 item $\mathbf{x}$ as estimated by the classifier, and by $E[x]$ we indicate the expected value of x; this corresponds to computing

$$\hat{p}_{Te}^{PCC}(c) = E[p_{Te}(\hat{c})] = \frac{1}{|Te|} \sum_{\mathbf{x} \in Te} p(c|\mathbf{x}) \quad (7)$$

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

The PCC method is dismissed as unsuitable in Forman (2005, 2008), 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 Bella et al. (2010) (where it is called "Probability Average") and Tang et al. (2010) but underperformed CC in the (much more extensive) experiments of Esuli and Sebastiani (2015). Interestingly enough, in the experiments of Gao and Sebastiani (2016), PCC proves the best performer, outperforming several among the methods that we discuss in this section.

Adjusted Classify and Count (ACC). Forman (2005, 2008) uses a further method which he calls "Adjusted Count" and which we will call (consistently with Esuli and Sebastiani 2015) *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

$$p_{Te}(\hat{c}_j) = \sum_{c_i, c_j \in \mathcal{C}} p_{Te}(\hat{c}_j|c_i) \cdot p_{Te}(c_i) \quad (8)$$

where $p_{Te}(\hat{c}_j|c_i)$ represents the fraction of test documents belonging to c_i that have been instead assigned to c_j 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 Eq. 8 (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 led most authors (see, e.g., Forman 2008) 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 Bella et al. (2010), 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 Eq. 8, $p_{Te}(\hat{c}_j)$ and $p_{Te}(\hat{c}_j|c_i)$ with their expected values. Equation 8 is thus transformed into

$$E[p_{Te}(\hat{c}_j)] = \sum_{c_i, c_j \in \mathcal{C}} E[p_{Te}(\hat{c}_j|c_i)] \cdot p_{Te}(c_i) \quad (9)$$

where

$$E\left[p_{Te}(\widehat{c}_j)\right] = \frac{1}{|Te|}\sum_{\mathbf{x}\in Te} p(c_j|\mathbf{x})$$
$$E\left[p_{Te}(\widehat{c}_j|c_i)\right] = \frac{1}{|Te_i|}\sum_{\mathbf{x}\in Te_i} p(c_j|\mathbf{x}) \quad (10)$$

and Te_i indicates the set of items in Te whose true class is c_i. Like for ACC, once the classifier has been trained and applied to Te, the quantity $E\left[p_{Te}(\widehat{c}_j)\right]$ can be observed, and the quantity $E\left[p_{Te}(\widehat{c}_j|c_i)\right]$ 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 values of $\widehat{p}_{Te}(c_i)$ 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 Forman (2005, 2008), for the same reasons for which PCC was also dismissed. Some experimental evidence seems instead in favor of PACC, since the experimental results published in Bella et al. (2010), Esuli and Sebastiani (2015), and Tang et al. (2010) indicate PACC to outperform all of CC, PCC, and ACC.

Expectation Maximization for Quantification (EMQ). EMQ, proposed by Saerens et al. (2002), is an instance of Expectation Maximization (Dempster et al. 1977), 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 12) 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.

All of the above methods require an underlying classifier that, given an item, predicts whether it belongs to class c or not (CC, ACC) or outputs a posterior probability of membership in c (PCC, PACC); any learning method for generating this classifier can be used. 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]), for PCC and PACC, these scores must be converted into true probabilities. If the score is a monotonically increasing function of the classifier's confidence in the fact that the item belongs to the class, this may be obtained by applying a logistic function. *Well-calibrated* probabilities (defined as the probabilities such that the prevalence $p_{\mathcal{S}}(c)$ of a class c in a set $\mathcal{S}$ is equal to $\sum_{\mathbf{x}\in\mathcal{S}} p(c|\mathbf{x})$) may be obtained by using a *generalized* logistic function.

Within the class of aggregative methods, a further distinction can be made between methods that use general-purpose learning algorithms, 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. All of the methods described so far belong to the former class; let us now look at two methods of the latter type.

SVMs Optimized for KLD **(SVM(KLD)).** SVM(KLD), proposed in Esuli and Sebastiani (2010b, 2015), is an instantiation of SVM-perf (Joachims 2005) that uses KLD as the loss to optimize. (In Joachims (2005), 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.) 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 "Evaluating Quantification" are). Instead of handling hypotheses h: $\mathcal{X} \rightarrow \mathcal{Y}$ that map an individual item (e.g., a tweet) $\mathbf{x}_i$ into an individual label $y_i \in \mathcal{Y}$, SVM-perf considers hypotheses $\overline{h} : \overline{\mathcal{X}} \rightarrow \overline{\mathcal{Y}}$ that map entire tuples of items (e.g., entire sets of tweets) $\overline{\mathbf{x}} = (\mathbf{x}_1, ..., \mathbf{x}_n)$ into tuples of labels $\overline{\mathbf{y}} = (y_1, ..., y_n)$. Instead of learning the traditional hypotheses of type

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}\cdot\mathbf{x} + b) \quad (11)$$

SVM-perf thus learns hypotheses of type

S

```
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                                                    */
1  $s \leftarrow 0$;
2  for $c \in \mathcal{C}$ do
3      $\hat{p}^{(s)}_{Te}(c) \leftarrow p_{Tr}(c)$;
4      for $\mathbf{x} \in Te$ do
5          $p^{(s)}(c|\mathbf{x}) \leftarrow p(c|\mathbf{x})$;
6      end
7  end
   /* Main Iteration Cycle                                              */
8  while stopping condition = false do
9      $s \leftarrow s + 1$;
10     for $c \in \mathcal{C}$ do
11         for $\mathbf{x} \in Te$ do
12             $p^{(s)}(c|\mathbf{x}) \leftarrow \dfrac{\dfrac{\hat{p}^{(s)}_{Te}(c)}{\hat{p}^{(0)}_{Te}(c)} \cdot p^{(0)}(c|\mathbf{x})}{\sum_{c \in \mathcal{C}} \dfrac{\hat{p}^{(s)}_{Te}(c)}{\hat{p}^{(0)}_{Te}(c)} \cdot p^{(0)}(c|\mathbf{x})}$
13         end
14         $\hat{p}^{(s)}_{Te}(c) \leftarrow \frac{1}{|Te|} \sum_{\mathbf{x} \in Te} p^{(s-1)}(c|\mathbf{x})$
15     end
16 end
   /* Generate output                                                   */
17 for $c \in \mathcal{C}$ do
18     $\hat{p}_{Te}(c) \leftarrow \hat{p}^{(s)}_{Te}(c)$
19 end
```

Sentiment Quantification of User-Generated Content, Algorithm 1 The EMQ algorithm (Saerens et al. 2002)

$$\overline{h}(\overline{\mathbf{x}}) = \arg \max_{\overline{y} \in \overline{\mathcal{Y}}} (\mathbf{w} \cdot \Psi(\overline{\mathbf{x}}, \overline{y})) \qquad (12)$$

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

$$\Psi(\overline{\mathbf{x}}, \overline{y}) = \sum_{i=1}^{n} \mathbf{x}_i y_i \qquad (13)$$

(the *joint feature map*) is a function that scores the pair of tuples $(\overline{\mathbf{x}}, \overline{y})$ according to how "compatible" $\overline{\mathbf{x}}$ and $\overline{y}$ 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 $\overline{\mathbf{x}}$ 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

$$\begin{aligned} &\arg \min_{\mathbf{w}, \xi_i \geq 0} \quad \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C \sum_{i=1}^{|Tr|} \xi_i \\ &\text{such that} \quad y'_i \left[\mathbf{w} \cdot \mathbf{x}'_i + b\right] \geq (1 - \xi_i) \\ &\qquad\qquad \text{for all } i \in \{1, \ldots, |Tr|\} \end{aligned} \qquad (14)$$

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

$$\begin{aligned} &\arg \min_{\mathbf{w},\, \xi_i \geq 0} \quad \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\xi \\ &\text{such that} \quad \mathbf{w}\cdot[\Psi(\overline{\mathbf{x}}',\overline{y}') - \Psi(\overline{\mathbf{x}}',\overline{y}) + b] \\ &\qquad \geq \Lambda(\overline{y}',\overline{y}) - \xi \ \text{ for all } \ \overline{y} \in \overline{\mathcal{Y}}/\overline{y}' \end{aligned} \tag{15}$$

where $(\overline{\mathbf{x}}', \overline{y}')$ 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 Λ explicitly appears in the optimization problem.

Note that the set of possible labels $\mathcal{Y}$ is equal to $\{c_i, \overline{c}_i\}$, where c_i is any of the classes we are concerned with and $\overline{c}_i$ is its complement; that is, SVM-perf can only deal with binary decisions, which makes SVM(KLD) apt for binary quantification only. If we want to tackle SLMC quantification with $|\mathcal{C}| > 2$ classes, we need to independently generate $|\mathcal{C}|$ predicted prevalences $\widehat{p}(c)$ for each $c \in \mathcal{C}$ via $|\mathcal{C}|$ instances of SVM (KLD) and then rescale these predicted prevalences so that they sum up to 1.

SVMs Optimized for Q (SVM(Q)). SVM(Q), originally proposed in Barranquero et al. (2015), is, like SVM(KLD), 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

$$Q_\beta(p,\widehat{p}) = \frac{(\beta^2+1)\Gamma_c(p,\widehat{p})\cdot\Gamma_q(p,\widehat{p})}{\beta^2\Gamma_c(p,\widehat{p}) + \Gamma_q(p,\widehat{p})} \tag{16}$$

where Γ_c and Γ_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_β measure coincides with Γ_c, while when β tends to $+\infty$, Q_β asymptotically tends to Γ_q. As a measure of classification gain, the authors use recall, while as a measure of quantification gain, they use (1 − RAE), where RAE is as defined in Eq. 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_β is unnecessary, since the presence of Γ_q in Q_β has the effect of ruling out anyway those hypotheses characterized by high recall and low precision/specificity (since these hypotheses are indeed penalized by Γ_q).

Key Applications

One of the first applications of quantification to the field of user-generated content was described in Hopkins and King (2010), where the authors estimate the prevalence of support for different political candidates from blog posts, using the quantification algorithm pioneered in King and Lu (2008).

Another such application is described in Gao and Sebastiani (2015), which discusses sentiment quantification of tweets; the authors show experimentally that SVM(KLD) outperforms a "classify and count" approach implemented via linear SVMs on several tweet sentiment datasets. Sentiment quantification of tweets is also the topic of Da San Martino et al. (2016), whose authors tackle quantification at the ordinal level (using a totally ordered set of five degrees of sentiment strength) via a hierarchical quantification approach. Tweet quantification is also one of the subjects of the recent SemEval Task 4 "Sentiment Analysis in Twitter" shared task (Nakov et al. 2016), where tweets are labeled according to the sentiment they convey toward a certain topic; Subtask D consists of a binary quantification task, and Subtask E consists of an ordinal quantification task (with tweets labeled according to a five-point scale).

More in general, fields such as the social sciences, political science, reputation management, and market research are obvious application fields for sentiment quantification of UGC. This derives from the fact that these fields tend to be *inherently* interested in aggregate (rather than individual) views of people's attitudes. For instance, social scientists study the distribution of a given phenomenon across a population of interest (sometimes breaking up the population according to age or geographical location or religion or others) and are hardly interested in whether a single individual is affected by the phenomenon. Broadly speaking, we might say that researchers in these fields are usually less interested in finding the needle in the haystack than in characterizing the haystack itself.

A large number of works in the disciplines mentioned in the previous paragraph use quantification "without knowingly doing so"; that is, unaware of the existence of methods specifically optimized for quantification, they use classification with the only goal of estimating class prevalences. In other words, these works use plain "classify and count," but the application they are looking at is an obvious candidate for "real" quantification techniques. Among them, Mandel et al. (2012) use tweet quantification in order to estimate, from a quantitative point of view, the emotional responses of the population (broken down according to location and gender) to a natural disaster; Esuli and Sebastiani (2010a) quantify responses to open-ended surveys for market research applications; O'Connor et al. (2010) analyze the correlation between public opinion as measured via tweet sentiment quantification and via traditional opinion polls; and Dodds et al. (2011) use tweet sentiment quantification in order to infer spatiotemporal happiness patterns.

Future Directions

Research in sentiment quantification of user-generated content is still at an early stage; what can we expect for the near future?

First of all, one aspect that would be worth investigating is how to generate sentiment-laden vectorial representations of UGC that are specific to quantification; up to now, the same representations used for sentiment classification have been used, and this may be suboptimal.

Second, areas that will likely see new developments are those of single-label multi-class quantification and ordinal quantification. Up to now, most research on quantification has focused on the binary case, but going beyond binary is important for sentiment analysis, where classes other than Positive and Negative are often important.

More in general, as awareness of quantification as a task on its own grows, we may expect fewer and fewer applied works to use simple "classify and count" and more and more of them to apply methods that have proven more accurate in the quantification literature. This in turn may encourage the investigation of new learning methods specific to quantification.

Cross-References

- ▶ Microblog Sentiment Analysis
- ▶ Semantic Sentiment Analysis of Twitter Data
- ▶ Sentiment Analysis in Social Media
- ▶ Sentiment Analysis of Microblogging Data
- ▶ Sentiment Analysis of Reviews
- ▶ Sentiment Analysis, Basic Tasks of
- ▶ Social Media Analysis for Monitoring Political Sentiment
- ▶ Twitter Microblog Sentiment Analysis
- ▶ User Sentiment and Opinion Analysis

References

Barranquero J, Diez J, del Coz JJ (2015) Quantification-oriented learning based on reliable classifiers. Pattern Recogn 48(2):591–604

Bella A, Ferri C, Hernandez-Orallo J, Ramirez-Quintana MJ (2010) Quantification via probability estimators. In: Proceedings of the 11th IEEE international conference on data mining (ICDM 2010), Sydney, pp 737–742

Cover TM, Thomas JA (1991) Elements of information theory. Wiley, New York

Csiszar I, Shields PC (2004) Information theory and statistics: a tutorial. Found Trends Commun Inf Theory 1(4):417–528

Da San Martino G, Gao W, Sebastiani F (2016) Ordinal text quantification. In: Proceedings of the 39th ACM conference on research and development in information retrieval (SIGIR 2016), Pisa, pp 937–940. https://doi.org/10.1145/2911451.2914749

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc B 39(1):1–38

Dodds PS, Harris KD, Kloumann IM, Bliss CA, Danforth CM (2011) Temporal patterns of happiness and information in a global social network: hedonometrics and Twitter. PLoS One 6(12). https://doi.org/10.1371/journal.pone.0026752

du Plessis MC, Sugiyama M (2012) Semi-supervised learning of class balance under class-prior change by distribution matching. In: Proceedings of the 29th international conference on machine learning (ICML 2012), Edinburgh

Esuli A, Sebastiani F (2010a) Machines that learn how to code open-ended survey data. Int J Mark Res 52(6):775–800

Esuli A, Sebastiani F (2010b) Sentiment quantification. IEEE Intell Syst 25(4):72–75

Esuli A, Sebastiani F (2015) Optimizing text quantifiers for multivariate loss functions. ACM Trans Knowl Discov Data 9(4), Article 27

Feldman R (2013) Techniques and applications for sentiment analysis. Commun ACM 56(4):82–89

Forman, G (2005) Counting positives accurately despite inaccurate classification. In: Proceedings of the 16th European conference on machine learning (ECML), Porto, pp 564–575

Forman G (2008) Quantifying counts and costs via classification. Data Min Knowl Disc 17(2):164–206

Gao W, Sebastiani F (2015) Tweet sentiment: from classification to quantification. In: Proceedings of the 7th international conference on advances in social network analysis and mining (ASONAM 2015), Paris, pp 97–104

Gao W, Sebastiani F (2016) From classification to quantification in tweet sentiment analysis. Soc Netw Anal Min 6(19):1–22

Gonzalez-Castro V, Alaiz-Rodriguez R, Alegre E (2013) Class distribution estimation based on the Hellinger distance. Inf Sci 218:146–164

Hopkins DJ, King G (2010) A method of automated nonparametric content analysis for social science. Am J Polit Sci 54(1):229–247

Joachims T. (2005) A support vector method for multivariate performance measures. In: Proceedings of the 22nd international conference on machine learning (ICML 2005), Bonn, pp 377–384

Kar P, Li S, Narasimhan H, Chawla S, Sebastiani F (2016) Online optimization methods for the quantification problem. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (KDD 2016), San Francisco, pp 1625–1634. https://doi.org/10.1145/2939672.2939832

King G, Lu Y (2008) Verbal autopsy methods with multiple causes of death. Stat Sci 23(1):78–91

Liu B (2012) Sentiment analysis and opinion mining. Morgan and Claypool Publishers, San Rafael

Mandel B, Culotta A, Boulahanis J, Stark D, Lewis B, Rodrigue J (2012) A demographic analysis of online sentiment during hurricane Irene. In: Proceedings of the NAACL/HLT workshop on language in social media, Montreal, pp 27–36

Nakov P, Ritter A, Rosenthal S, Sebastiani F, Stoyanov V (2016) SemEval-2016 Task 4: sentiment analysis in Twitter. In: Proceedings of the 10th international workshop on semantic evaluation (SemEval 2016), San Diego, pp 1–18

O'Connor B, Balasubramanyan R, Routledge BR, Smith NA (2010) From tweets to polls: linking text sentiment to public opinion time series. In: Proceedings of the 4th AAAI conference on weblogs and social media (ICWSM 2010), Washington, DC

Pang B, Lee L (2008) Opinion mining and sentiment analysis. Found Trends Inf Retr 2(1/2):1–135

Saerens M, Latinne P, Decaestecker C (2002) Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. Neural Comput 14(1):21–41

Tang L, Gao H, Liu H (2010) Network quantification despite biased labels. In: Proceedings of the 8th workshop on mining and learning with graphs (MLG 2010), Washington, DC, pp 147–154

Tang D, Wei F, Yang N, Zhou M, Liu T, Qin B (2014) Learning sentiment-specific word embedding for Twitter sentiment classification. In: Proceedings of the 52nd annual meeting of the Association for Computational Linguistics (ACL 2014), Baltimore, pp 1555–1565

Vapnik V (1998) Statistical learning theory. Wiley, New York

[redacted]