A Recurrent Neural Network for Sentiment Quantification

Andrea Esuli, Alejandro Moreo Fernández, Fabrizio Sebastiani
Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
firstname.lastname@isti.cnr.it

ABSTRACT

Quantification is a supervised learning task that consists in predicting, given a set of classes C and a set D of unlabelled items, the prevalence (or relative frequency) $p_c(D)$ of each class $c \in C$ in D. Quantification can in principle be solved by classifying all the unlabelled items and counting how many of them have been attributed to each class. However, this "classify and count" approach has been shown to yield suboptimal quantification accuracy; this has established quantification as a task of its own, and given rise to a number of methods specifically devised for it. We propose a recurrent neural network architecture for quantification (that we call *QuaNet*) that observes the classification predictions to learn higher-order "quantification embeddings", which are then refined by incorporating quantification predictions of simple classify-and-count-like methods. We test *QuaNet* on sentiment quantification on text, showing that it substantially outperforms several state-of-the-art baselines.

KEYWORDS

Quantification, Neural Networks, Deep Learning, Sentiment Analysis, Opinion Mining

ACM Reference Format:
Andrea Esuli, Alejandro Moreo Fernández, Fabrizio Sebastiani. 2018. A Recurrent Neural Network for Sentiment Quantification. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3269206.3269287

1 INTRODUCTION

Quantification (also known as "class prior estimation") is a supervised learning task that consists of predicting, given a set of classes C and a set D (a *sample*) of unlabelled items drawn from some domain $\mathcal{D}$, the prevalence (i.e., relative frequency) $p_c(D)$ of each class $c \in C$ in D. Put it another way, given an unknown distribution $p_C(D)$ of the members of D across C (the *true distribution*), quantification consists in generating a *predicted distribution* $\hat{p}_C(D)$ that approximates $p_C(D)$ as accurately as possible [8]. Quantification is important for many application fields characterised by an interest in aggregate (rather than individual) data, such as the social sciences, market research, political science, and epidemiology.

In principle, quantification can be trivially solved via classification, i.e., by training a classifier h using training data labelled according to C, classifying the unlabelled data in D via h, and counting, for each $c \in C$, how many items in D have been attributed to c (the "classify and count" method). However, research has conclusively shown that this approach leads to suboptimal quantification accuracy [1, 2, 5, 6]. Simply put, the reason is that classifiers are typically trained to minimize classification error, which is by and large proportional to $(FP + FN)$, while a good quantifier should be trained to minimise quantification error, which is by and large proportional to $|FP - FN|$ (where TP, FP, FN, TN denote the usual counts from a binary contingency table).

In this paper we tackle quantification in a binary setting, and propose a recurrent neural network architecture (that we call *QuaNet*) that observes the classification predictions to learn higher-order "quantification embeddings", which are then refined by incorporating quantification predictions of simple classify-and-count-like methods. We apply this method to the problem of performing quantification by sentiment (i.e., C={Positive,Negative}) on text. The aspect of sentiment is of special interest for quantification [3, 7], since quantification (as applied to e.g., posts from social networks, or product reviews) is a key tool to monitor opinion trends on objects of interest (e.g., products, policies, or political candidates).

To the best of our knowledge, QuaNet is the first deep learning approach to sentiment quantification proposed in the literature.

2 RELATED WORK

The simplest quantification method (and the one that acts as a lower-bound baseline for all quantification methods) is the above-mentioned *classify and count* (CC), which comes down to computing

$$\hat{p}_c^{CC}(D) = \frac{|\{\mathbf{x} \in D | h(\mathbf{x}) = c\}|}{|D|} = \frac{TP_b + FP_b}{|D|} \tag{1}$$

where the b subscript indicates that the values are counts of binary predictions (the reason for specifying this will become clear later). It is easy to verify that the relation between $\hat{p}_c^{CC}(D)$ and the true prevalence $p_c(D)$ is given, in the binary case,[1] by

$$p_c(D) = \frac{\hat{p}_c^{CC}(D) - fpr_b}{tpr_b - fpr_b} \tag{2}$$

where tpr_b and fpr_b denote the *true positive rate* $tpr_b = \frac{TP_b}{TP_b+FN_b}$ and *false positive rate* $fpr_b = \frac{FP_b}{FP_b+TN_b}$ of h on D. This relationship justifies the *adjusted classify and count* (ACC) quantification method [6], which consists in computing

$$\hat{p}_c^{ACC}(D) = \frac{\hat{p}_c^{CC}(D) - \hat{fpr}_b}{\hat{tpr}_b - \hat{fpr}_b} \tag{3}$$

[1] Equation 2 is easily extended to the multiclass case (see e.g., [7]). For brevity, in the rest of the paper we will exclusively deal with the binary case.

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3269287

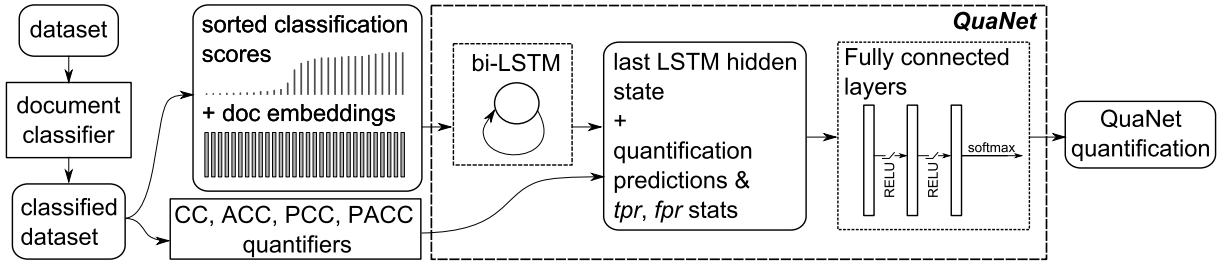

Figure 1: Architecture of the QuaNet quantification system.

where $\hat{tpr}_b$ and $\hat{fpr}_b$ are estimates of tpr_b and fpr_b computed either on a held-out set or via k-fold cross-validation.

CC and ACC use the binary predictions generated by classifier h. Since most classifiers can be configured to output "soft predictions" in the form of posterior probabilities $\Pr(c|\mathbf{x})$ (from which binary predictions are obtained via thresholding),[2] and since posterior probabilities contain richer information than binary predictions, one can generate probabilistic versions of the CC and ACC methods [2] based on "soft contingency tables", i.e., contingency tables where counts TP_b, FP_b, FN_b, TN_b are replaced by *expected counts* TP_s, FP_s, FN_s, TN_s (subscript s indicating that the corresponding values derive from soft predictions) obtained from the

		True	
		Pos	Neg
Predicted	Pos	$TP_s = \sum_{\mathbf{x}\in c,D} \Pr(c\|\mathbf{x})$	$FP_s = \sum_{\mathbf{x}\in \overline{c},D} \Pr(c\|\mathbf{x})$
	Neg	$FN_s = \sum_{\mathbf{x}\in c,D} (1-\Pr(c\|\mathbf{x}))$	$TN_s = \sum_{\mathbf{x}\in \overline{c},D} (1-\Pr(c\|\mathbf{x}))$

It is thus possible to define "soft" variants of tpr_b and fpr_b as $tpr_s = \frac{TP_s}{TP_s+FN_s}$ and $fpr_s = \frac{FP_s}{FP_s+TN_s}$. One can then define probabilistic versions of the quantification methods defined by Equations 1 and 3, namely, *probabilistic classify and count* (PCC), defined by

$$\hat{p}_c^{PCC}(D) = \frac{\sum_{\mathbf{x}\in D} \Pr(c|\mathbf{x})}{|D|} = \frac{TP_s + FP_s}{|D|} \tag{4}$$

and *probabilistic adjusted classify and count* (PACC), defined by

$$\hat{p}_c^{PACC}(D) = \frac{\hat{p}_c^{PCC}(D) - \hat{fpr}_s}{\hat{tpr}_s - \hat{fpr}_s} \tag{5}$$

The four quantification methods defined by Equations 1, 3, 4, 5, use the output of general-purpose classifiers. A radically different approach is taken in [5], which suggests to apply Equation 1 to the output of a classifier specifically designed for optimizing quantification accuracy. Since any measure of quantification accuracy is inherently nonlinear and multivariate, [5] suggests using the SVM_{perf} structured output learner [9], which can indeed optimize any nonlinear and multivariate loss function that can be expressed in terms of a contingency table. Three quantification methods have emerged in the literature as a result of this proposal, namely, (i) SVM^{KLD} [5], which minimizes the *Kullback-Leibler divergence* (*KLD* – see Equation 8), (ii) SVM^{NKLD} [4], which minimizes a normalized version of *KLD*, and (iii) SVM^{Q} [1], which minimizes the harmonic mean of a classification-oriented loss (recall) and a quantification-oriented loss (*RAE* – see Equation 7). These methods still represent the state of the art in terms of quantification effectiveness.

Yet another approach to quantification is exemplified by the EMQ method of [12], which consists in generating a probabilistic classifier while exploiting the EM algorithm to iteratively shift the estimation of $p_C(D)$ from the one of the training set to one that maximizes the likelihood on the test data.

In Section 4 we will use all the quantification methods presented in this section as baselines against which to compare our approach.

3 DEEP LEARNING FOR QUANTIFICATION

We propose a new deep learning method for quantification, that we dub QuaNet. The QuaNet architecture (see Figure 1) is composed of a main recurrent component (a bidirectional LSTM) that, given a class c and a classifier h which returns posterior probabilities $\Pr(c|\mathbf{x})$, receives as input a list L of pairs $\langle \Pr(c|\mathbf{x}), \vec{\mathbf{x}} \rangle$, one for each $\mathbf{x} \in D$. Here, $\vec{\mathbf{x}}$ is the document embedding for $\mathbf{x}$, and L is sorted by the value of $\Pr(c|\mathbf{x})$. The intuition is that the recurrent network "learns to count" the Positive and Negative examples by observing them. The rationale of *sorting* the list of posterior probabilities is that by observing the ordered sequence of $\Pr(c|\mathbf{x})$ values the network can learn to recognize the switch point between Negative and Positive documents. The document embedding $\vec{\mathbf{x}}$ associated to each score is a vectorial representation of the content of the document. The embedding can be exploited by the network for learning to weight the contribution of each document to the final estimate.

The vector returned by the LSTM can be considered a "quantification embedding", i.e., a dense, multi-dimensional representation of the information relevant to quantification observed from the input data. This vector is then concatenated to a short vector that contains (i) the $\hat{p}_c^{CC}(D)$, $\hat{p}_c^{ACC}(D)$, $\hat{p}_c^{PCC}(D)$ and $\hat{p}_c^{PACC}(D)$ predictions, and (ii) the tpr_b, fpr_b, tpr_s, fpr_s statistics, i.e., a set of aggregate values that are very cheap to determine from the classifier h.[3] The resulting vector passes through the second component of QuaNet, that consists of a number of fully connected layers, with ReLU activation, which "correct" the quantification embedding from the

[2] If a classifier natively outputs classification scores that are not probabilities, the former can be converted into the latter via "probability calibration"; see e.g., [11].

[3] In additional experiments, not reported in this paper for reasons of space, we have performed a number of ablation experiments which indeed confirmed that all these inputs to QuaNet do contribute to improving the quantification accuracy of QuaNet.

LSTM by exploiting the explicit quantification-related statistics received from the classifier. The final quantification embedding from this stage is then converted into the $\hat{p}_c^{QuaNet}(D)$ prediction by a final layer of size 2 with softmax activation.

In principle, QuaNet could exploit (along those coming from CC, ACC, PCC, PACC) quantification predictions from any method, and the richer the set of predictions the more information the network would have for returning an accurate output. However, it is reasonable to use only quantification predictions that can be computed with minimal computational cost; for this reason we stick to just the above mentioned predictions that are computed directly from the output of classifier h, while the tpr_b, fpr_b, tpr_s, fpr_s values needed for computing $\hat{p}_c^{ACC}(D)$ and $\hat{p}_c^{PACC}(D)$ are easily determined by applying h to a validation set.

In our QuaNet implementation[4] the LSTM cells have 64 hidden dimensions; the output of the LSTM (a vector of size 128, since the LSTM is bidirectional) is concatenated with the eight quantification statistics from the classifier to form a vector with 136 dimensions that goes through two dense layers of 1,024 and 512 dimensions, each with a ReLU activation and a 0.5 dropout layer, ending with a layer of size two with softmax activation.

4 EXPERIMENTS

We run experiments on three sentiment classification datasets: (i) IMDB, the popular *Large Movie Review Dataset* [10]; (ii) Kindle, a set of reviews of Kindle e-book readers, and (iii) HP, a set of reviews of the books from the Harry Potter series. The IMDB dataset consists of 25,000 training and 25,000 test documents. We crawled the Kindle and HP datasets ourselves from the Amazon website; we consider all 5-stars and 4-stars reviews as Positive and all 1-star and 2-stars reviews as Negative, discarding all 3-stars reviews.[5] The Kindle dataset consists of the 25,413 reviews from Aug 2008 to Jul 2011; we take the 3,821 reviews from the first three months as the training set, leaving 21,592 reviews for testing. The HP corpus consists of the 27,934 reviews from 1998 to 2011; we use the 9,533 reviews from the first three years for training and the remaining 18,401 documents for testing. From each set of training data we randomly select 60% for training purposes, leaving the remaining 40% for parameter optimization. The IMDB, Kindle, and HP datasets are examples of balanced, imbalanced, and severely imbalanced datasets, since their training set prevalence of the Positive class is 50.0%, 91.7% 98.2%, respectively.

In order to evaluate the effectiveness of a quantifier over a wide spectrum of test prevalences we follow the experimental protocol of Forman [6], which consists of randomly undersampling one of the two classes in the test set in order to obtain a sample with specified class prevalences. We extract samples with a prevalence of the Positive class in the set $\{0.01, 0.05, 0.10, \ldots, 0.90, 0.95, 0.99\}$.[6] For each of these 21 values we generate 100 random samples of 500 test documents each; we thus report the averaged quantification error across $21 \times 100 = 2100$ test samples. As the measures of quantification error we use *Absolute Error* (*AE*), *Relative Absolute Error* (*RAE*), and *Kullback-Leibler Divergence* (*KLD*) [8], defined as

$$AE(p, \hat{p}) = \frac{1}{|C|} \sum_{c \in C} |\hat{p}(c) - p(c)| \tag{6}$$

$$RAE(p, \hat{p}) = \frac{1}{|C|} \sum_{c \in C} \frac{|\hat{p}(c) - p(c)|}{p(c)} \tag{7}$$

$$KLD(p, \hat{p}) = \sum_{c \in C} p(c) \log_e \frac{p(c)}{\hat{p}(c)} \tag{8}$$

We apply the above formulas individually on each test sample and then take the mean across the resulting 2,100 values. We compare our approach against all the methods discussed in Section 2.

We adopt an LSTM network also as the document classifier[7] since this architecture naturally provides, for each input document, both a classification score and an embedded representation as the last and next-to-last layers, respectively. In this LSTM each unique word is first transformed into a 100-dimensional embedding, which is then processed by the LSTM cells with 128 hidden dimensions. The output of the LSTM is followed by two fully connected layers (of 1,024 and 100 dimensions, resp.), each followed by a ReLU activation and a 0.5 dropout layer. Another last fully connected projection takes the 100-dimensional document embedding to produce two outputs, via a softmax activation, which eventually represent the confidence scores for the Positive and Negative classes.

We use this LSTM-based classifier both for QuaNet and also for the CC, ACC, PCC, and PACC baselines, for a direct and fair comparison. For SVM^{KLD}, SVM^{NKLD}, and SVM^{Q}, we use Joachims' SVM_{perf} implementation [9], instantiated with the corresponding loss functions. For each of these methods we choose the C parameter in the range $C = 10^i$, with $i \in [-3, -2, \ldots 2, 3]$, that deliver the lowest quantification error on the validation set. As the base for EMQ we use the `scikit-learn` implementation of Multinomial Naïve Bayes (MNB); we choose the best value for its α parameter via grid search on $\alpha = 10^i$ with $i \in [-4, -2, \ldots 2, 4]$. As the loss for both neural networks (i.e., classifier and quantifier) we use the Mean Square Error (*MSE*) of the predictions, minimized via stochastic gradient descent; we apply the Adam updating rule, with learning rate $lr = 10^{-4}$ and weight decay $wd = 10^{-4}$. We set the batch size to 100 and the maximum number of iterations to 20,000, but we apply an early stopping criterion after 20 validations that do not show any improvement (we validate every 100 iterations). Classification batches are composed of documents randomly sampled from the dataset. Quantification batches contain quantification examples that uniformly cover the entire prevalence range.

4.1 Results

Figure 2 displays the performance of the different quantifiers. Each point on a line denotes the average estimated prevalence across the 100 different random trials, while coloured bands indicate the standard deviation. The pale blue circle indicates the training set prevalence of the Positive class; the (ideal) perfect quantifier is represented by the diagonal.

[4] The implementations of QuaNet and of all the baseline methods are available at https://github.com/HLT-ISTI/QuaNet

[5] These datasets are available at http://hlt.isti.cnr.it/quantification/

[6] We do not use 0 or 1 prevalences (and use instead 0.01 and 0.99). This is due to the fact that, when $p_c(D)$ is 0 or 1, *RAE* and *KLD* (i) are undefined, and thus require smoothing to be defined, and even when smoothed, they are numerically unstable.

[7] Note that QuaNet is independent of the classifier, and thus the combination of any classifier that outputs confidence scores with any method producing document embeddings could in principle represent valid inputs for QuaNet. Also, the classifier and QuaNet are trained and optimized separately.

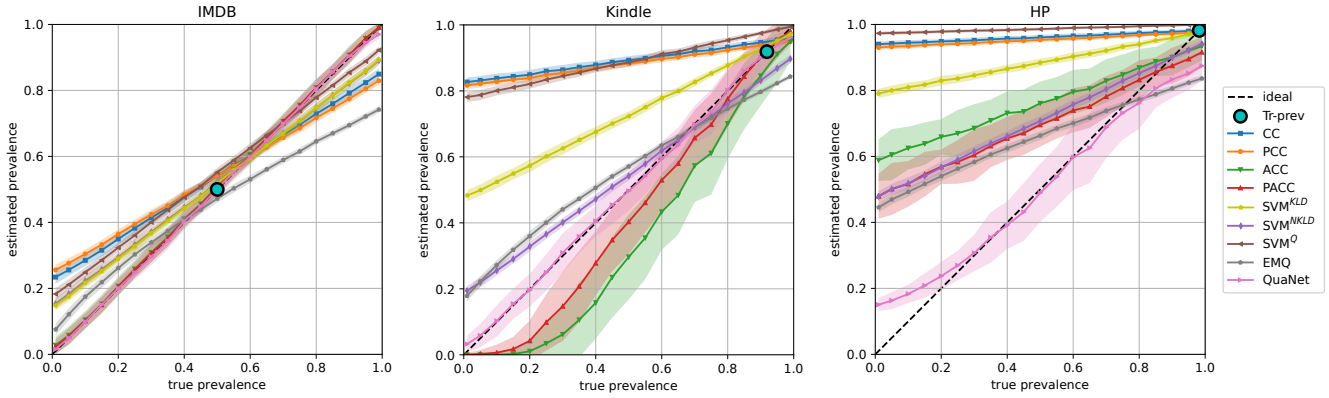

Figure 2: Prevalences of the Positive class predicted by the various methods, against the corresponding true prevalences.

	IMDB						Kindle						HP					
	AE		RAE		KLD		AE		RAE		KLD		AE		RAE		KLD	
CC	0.096	(+421%)	1.193	(+1008%)	0.044	(+1419%)	0.417	(+585%)	5.805	(+1083%)	0.717	(+3288%)	0.476	(+379%)	6.487	(+526%)	1.346	(+2789%)
ACC	0.021	(+15%)	0.144	(+34%)	0.011	(+273%)	0.160	(+161%)	0.884†	(+80%)	0.839†	(+3866%)	0.245	(+146%)	2.919	(+181%)	0.400	(+759%)
PCC	0.107	(+483%)	1.336	(+1141%)	0.053	(+1704%)	0.407	(+567%)	5.704	(+1062%)	0.661	(+3022%)	0.469	(+371%)	6.411	(+519%)	1.202	(+2480%)
PACC	0.019††	(+1%)	0.126†	(+17%)	0.010	(+233%)	0.138	(+127%)	0.688††	(+40%)	0.736†	(+3378%)	0.205	(+106%)	2.391†	(+130%)	0.267	(+473%)
SVM^{KLD}	0.065	(+253%)	0.890	(+727%)	0.025	(+741%)	0.228	(+273%)	3.186	(+549%)	0.171	(+708%)	0.385	(+287%)	5.298	(+411%)	0.543	(+1065%)
SVM^{NKLD}	0.068	(+269%)	0.938	(+771%)	0.026	(+795%)	0.080	(+32%)†	1.192	(+142%)	0.033†	(+58%)	0.217	(+118%)	3.146	(+203%)	0.160	(+244%)
SVM^{Q}	0.075	(+305%)	1.127	(+947%)	0.029	(+907%)	0.387	(+535%)	5.246	(+969%)	0.548	(+2489%)	0.486	(+389%)	6.560	(+533%)	1.643	(+3427%)
EMQ	0.094	(+411%)	0.529	(+391%)	0.045	(+1428%)	0.104	(+70%)	1.196	(+143%)	0.046	(+115%)	0.200	(+101%)	2.912	(+181%)	0.140	(+201%)
QuaNet	**0.018**		**0.108**		**0.003**		**0.061**		**0.491**		**0.021**		**0.099**		**1.036**		**0.047**	

Table 1: Average errors of the quantification methods. Symbols † and †† denote the methods (if any) whose score is *not* statistically significantly different from the best one (in boldface) at $\alpha = 0.05$ (†) or at $\alpha = 0.005$ (††) according to a two-tailed paired t-test measured on 10 runs of the experimental setup. Percentages indicate increase in error with respect to the best method.

CC and PCC are adversely influenced by training class prevalences, and this is especially true for the imbalanced and severely imbalanced Kindle and HP. ACC and PACC show superior performance to their un-adjusted counterparts CC and PCC, as especially clear on IMDB; yet they still leave room for improvement in the more difficult cases of Kindle (where both tend to underestimate) and HP (where both tend to overestimate). The corrections tend to produce higher variances in the estimations (as shown by the broader coloured bands). SVM-based methods are more stable, producing very low variations across different samples; SVM^{NKLD} shows better performance than SVM^{KLD} and SVM^{Q}, especially on the Kindle and HP datasets. The plots show that QuaNet is consistently the best quantifier across all datasets.

Table 1 reports the quantification results for the three evaluation metrics, averaged across 10 runs (the same 10 training/validation random splits are used consistently across all methods). QuaNet produces much lower quantification error than any baseline, for all three error measures and all three datasets. PACC shows comparable performance (in a statistically significant sense) to QuaNet only in a few cases, e.g., on IMDB for *AE* and in Kindle for *RAE*.

ACKNOWLEDGMENTS

The first author thanks NVidia corp. for granting a Titan X GPU.

REFERENCES

[1] José Barranquero, Jorge Díez, and Juan José del Coz. 2015. Quantification-oriented learning based on reliable classifiers. *Pattern Recognition* 48, 2 (2015), 591–604. https://doi.org/10.1016/j.patcog.2014.07.032

[2] Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. 2010. Quantification via Probability Estimators. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM 2010)*. Sydney, AU, 737–742. https://doi.org/10.1109/icdm.2010.75

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

[4] Andrea Esuli and Fabrizio Sebastiani. 2014. Explicit Loss Minimization in Quantification Applications (Preliminary Draft). In *Proceedings of the 8th International Workshop on Information Filtering and Retrieval (DART 2014)*. Pisa, IT, 1–11.

[5] Andrea Esuli and Fabrizio Sebastiani. 2015. Optimizing Text Quantifiers for Multivariate Loss Functions. *ACM Transactions on Knowledge Discovery and Data* 9, 4 (2015), Article 27. https://doi.org/10.1145/2700406

[6] George Forman. 2008. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery* 17, 2 (2008), 164–206. https://doi.org/10.1007/s10618-008-0097-y

[7] Wei Gao and Fabrizio Sebastiani. 2016. From Classification to Quantification in Tweet Sentiment Analysis. *Social Network Analysis and Mining* 6, 19 (2016), 1–22. https://doi.org/10.1007/s13278-016-0327-z

[8] Pablo González, Alberto Castaño, Nitesh V. Chawla, and Juan José del Coz. 2017. A Review on Quantification Learning. *Comput. Surveys* 50, 5 (2017), 74:1–74:40. https://doi.org/10.1145/3117807

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

[10] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*. Portland, US, 142–150.

[11] John C. Platt. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classifiers*, Alexander Smola, Peter Bartlett, Bernard Schölkopf, and Dale Schuurmans (Eds.). The MIT Press, Cambridge, MA, 61–74.

[12] Marco Saerens, Patrice Latinne, and Christine Decaestecker. 2002. Adjusting the Outputs of a Classifier to New a Priori Probabilities: A Simple Procedure. *Neural Computation* 14, 1 (2002), 21–41.