A Detailed Overview of LeQua@CLEF 2022: Learning to Quantify

Andrea Esuli, Alejandro Moreo, Fabrizio Sebastiani and Gianluca Sperduti

Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy

Abstract

LeQua 2022 is a new lab for the evaluation of methods for "learning to quantify" in textual datasets, i.e., for training predictors of the relative frequencies of the classes of interest $\mathcal{Y} = \{y_1, ..., y_n\}$ in sets of unlabelled textual documents. While these predictions could be easily achieved by first classifying all documents via a text classifier and then counting the numbers of documents assigned to the classes, a growing body of literature has shown this approach to be suboptimal, and has proposed better methods. The goal of this lab is to provide a setting for the comparative evaluation of methods for learning to quantify, both in the binary setting and in the single-label multiclass setting; this is the first time that an evaluation exercise solely dedicated to quantification is organized. For both the binary setting and the single-label multiclass setting, data were provided to participants both in ready-made vector form and in raw document form. In this overview article we describe the structure of the lab, we report the results obtained by the participants in the four proposed tasks and subtasks, and we comment on the lessons that can be learned from these results.

Keywords

Quantification, Learning to quantify, Supervised class prevalence estimation, Prior estimation

1. Learning to Quantify

In a number of applications involving classification, the final goal is not determining which class (or classes) individual unlabelled items (e.g., textual documents, images, or other) belong to, but estimating the *prevalence* (or "relative frequency", or "prior probability", or "prior") of each class $y \in \mathcal{Y} = \{y_1, ..., y_n\}$ in the unlabelled data. Estimating class prevalence values for unlabelled data via supervised learning is known as *learning to quantify* (LQ) (or *quantification*, or *supervised prevalence estimation*) [1, 2].

LQ has several applications in fields (such as the social sciences, political science, market research, epidemiology, and ecological modelling) which are inherently interested in characterising *aggregations* of individuals, rather than the individuals themselves; disciplines like the ones above are usually *not* interested in finding the needle in the haystack, but in characterising the haystack. For instance, in most applications of tweet sentiment classification we are not

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy
andrea.esuli@isti.cnr.it (A. Esuli); alejandro.moreo@isti.cnr.it (A. Moreo); fabrizio.sebastiani@isti.cnr.it (F. Sebastiani); gianluca.sperduti@isti.cnr.it (G. Sperduti)
0000-0002-5725-4322 (A. Esuli); 0000-0002-0377-1025 (A. Moreo); 0000-0003-4221-6427 (F. Sebastiani); 0000-0002-4287-8968 (G. Sperduti)

concerned with estimating the true class (e.g., Positive, or Negative, or Neutral) of individual tweets. Rather, we are concerned with estimating the relative frequency of these classes in the set of unlabelled tweets under study; or, put in another way, we are interested in estimating as accurately as possible the true distribution of tweets across the classes.

It is by now well known that performing quantification by classifying each unlabelled instance and then counting the instances that have been attributed to the class (the "classify and count" method) usually leads to suboptimal quantification accuracy (see e.g., [2, 3, 4, 5, 6, 7, 8, 9, 10]); this may be seen as a direct consequence of "Vapnik's principle" [11], which states

> 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 our case, the problem to be solved directly is quantification, while the more general intermediate problem is classification.

One reason why "classify and count" is suboptimal is that many application scenarios suffer from *distribution shift*, the phenomenon according to which the distribution across the classes $y_1, ..., y_n$ in the sample (i.e., set) σ of *unlabelled* documents may substantially differ from the distribution across the classes in the labelled *training* set L; distribution shift is one example of *dataset shift* [12, 13], the phenomenon according to which the joint distributions $p_L(\mathbf{x}, y)$ and $p_\sigma(\mathbf{x}, y)$ differ. The presence of distribution shift means that the well-known IID assumption, on which most learning algorithms for training classifiers hinge, does not hold. In turn, this means that "classify and count" will perform suboptimally on sets of unlabelled items that exhibit distribution shift with respect to the training set, and that the higher the amount of shift, the worse we can expect "classify and count" to perform.

As a result of the suboptimality of the "classify and count" method, LQ has slowly evolved as a task in its own right, different (in goals, methods, techniques, and evaluation measures) from classification [2]. The research community has investigated methods to correct the biased prevalence estimates of general-purpose classifiers [3, 4, 5], supervised learning methods specially tailored to quantification [6, 7, 8, 9, 10], evaluation measures for quantification [14, 15], and protocols for carrying out this evaluation. Specific applications of LQ have also been investigated, such as sentiment quantification [16, 17, 18, 19], quantification in networked environments [20], or quantification for data streams [21]. For the near future it is easy to foresee that the interest in LQ will increase, due (a) to the increased awareness that "classify and count" is a suboptimal solution when it comes to prevalence estimation, and (b) to the fact that, with larger and larger quantities of data becoming available and requiring interpretation, in more and more scenarios we will only be able to afford to analyse these data at the aggregate level rather than individually.

2. The rationale for LeQua 2022

The LeQua 2022 lab (https://lequa2022.github.io/) at CLEF 2022 has a "shared task" format; it is a new lab, in two important senses:

- No labs on LQ have been organized before at CLEF conferences.
- Even outside the CLEF conference series, quantification has surfaced only episodically in previous shared tasks. The first such shared task was SemEval 2016 Task 4 "Sentiment Analysis in Twitter" [22], which comprised a *binary quantification* subtask and an *ordinal quantification* subtask (these two subtasks were offered again in the 2017 edition). Quantification also featured in the Dialogue Breakdown Detection Challenge [23], in the Dialogue Quality subtasks of the NTCIR-14 Short Text Conversation task [24], and in the NTCIR-15 Dialogue Evaluation task [25]. However, quantification was never the real focus of these tasks. For instance, the real focus of the tasks described by Nakov et al. [22] was sentiment analysis on Twitter data, to the point that almost all participants in the quantification subtasks used the trivial "classify and count" method, and focused, instead of optimising the quantification component, on optimising the sentiment analysis component, or on picking the best-performing learner for training the classifiers used by "classify and count". Similar considerations hold for the tasks discussed in [23, 24, 25].

This is the first time that a shared task whose explicit focus is quantification is organized. A lab on this topic was thus sorely needed, because the topic has great applicative potential, and because a lot of research on this topic has been carried out without the benefit of the systematic experimental comparisons that only shared tasks allow.

We expect the quantification community to benefit significantly from this lab. One of the reasons is that this community is spread across different fields, as also witnessed by the fact that work on LQ has been published in a scattered way across different areas, e.g., information retrieval [5, 7, 16], data mining [4, 8], machine learning [26, 27], statistics [28], or in the areas to which these techniques get applied [17, 29, 30]. In their papers, authors often use as baselines only the algorithms from their own fields; one of the goals of this lab was thus to pull together people from different walks of life, and to generate cross-fertilisation among the respective sub-communities.

While quantification is a general-purpose machine learning / data mining task that can be applied to any type of data, in this lab we focus on its application to data consisting of textual documents.

3. Setting up LeQua 2022

In quantification, a *data item* (usually represented as $\mathbf{x}$) is the individual unit of information; for instance, a textual document, an image, a video, are examples of data items. In LeQua 2022, as data items we use textual *documents* (and, more specifically, product reviews). A document $\mathbf{x}$ has a *label*, i.e., it belongs to a certain class $y \in \mathcal{Y} = \{y_1, ..., y_n\}$; in this case we say that y is the label of $\mathbf{x}$. In LeQua 2022, classes are either merchandise classes for products, or sentiment classes for reviews (see Section 3.4 for more).

Some documents are such that their label is known to the quantification algorithm, and are thus called *labelled items*; we typically use them as training examples for the quantifier-training algorithm. Some other documents are such that their label is unknown to the quantifier-training algorithm and to the trained quantifier, and are thus called *unlabelled items*; for testing purposes we use documents whose label we hide to the quantifier. Unlike a classifier, a quantifier must

not predict labels for individual documents, but must predict *prevalence values* for *samples* (i.e., sets) of unlabelled documents; a prevalence value for a class y and a sample σ is a number in [0,1] such that the prevalence values for the classes in $\mathcal{Y} = \{y_1, ..., y_n\}$ sum up to 1. Note that when, in the following, we use the term "label", we always refer to the label of an individual document (and not of a sample of documents; samples do not have labels, but prevalence values for classes).

3.1. Tasks

Two tasks (T1 and T2) were offered within LeQua 2022, each admitting two subtasks (A and B).

In Task T1 (the *vector task*) participant teams were provided with vectorial representations of the (training / development / test) documents. This task was offered so as to appeal to those participants who are not into *text* learning, since participants in this task did not need to deal with text preprocessing issues. Additionally, this task allowed the participants to concentrate on optimising their quantification methods, rather than spending time on optimising the process for producing vectorial representations of the documents.

In Task T2 (the *raw documents task*), participant teams were provided with the raw (training / development / test) documents. This task was offered so as to appeal to those participants who wanted to deploy end-to-end systems, or to those who wanted to also optimise the process for producing vectorial representations of the documents (possibly tailored to the quantification task).

The two subtasks of both tasks were the *binary quantification subtask* (T1A and T2A) and the *single-label multiclass quantification subtask* (T1B and T2B); in both subtasks each document belongs to only one of the classes of interest $y_1, ..., y_n$, with $n = 2$ in T1A and T2A and $n > 2$ in T1B and T2B.

The four subtasks conceptually form a 2×2 grid, as illustrated in the following table.

	Binary (by sentiment)	**Multiclass** (by topic)
Vector	T1A	T1B
Raw Documents	T2A	T2B

For each subtask in { T1A, T1B, T2A, T2B }, participant teams were required not to use (training / development / test) documents other than those provided for that subtask. In particular, participants were explicitly advised against using any document from either T2A or T2B in order to solve either T1A or T1B.

3.2. The evaluation protocol

As the protocol for generating the test samples on which the quantifiers will be tested we adopt the so-called *artificial prevalence protocol* (APP), which is by now a standard protocol for generating the datasets to be used in the evaluation of quantifiers. Using the APP consists of taking the test set U of unlabelled data items, and extracting from it a number of subsets (the *test samples*), each characterised by a *predetermined* vector $(p_\sigma(y_1), ..., p_\sigma(y_n))$ of prevalence values, where $y_1, ..., y_n$ are the classes of interest. In other words, for extracting a test sample σ, we generate a vector of prevalence values, and randomly select documents from U accordingly (i.e., by class-conditional random selection of documents until the desired class prevalence values are obtained).[1]

The goal of the APP is to generate samples characterised by widely different vectors of prevalence values; this is meant to test the robustness of a *quantifier* (i.e., of an estimator of class prevalence values) in confronting class prevalence values possibly different (or very different) from the ones of the set it has been trained on. For doing this we draw the vectors of class prevalence values uniformly at random from the set of all legitimate such vectors, i.e., from the *unit* $(n-1)$*-simplex* of all vectors $(p_\sigma(y_1), ..., p_\sigma(y_n))$ such that $p_\sigma(y_i) \in [0, 1]$ for all $y_i \in \mathcal{Y}$ and $\sum_{y_i \in \mathcal{Y}} p_\sigma(y_i) = 1$. For this we use the Kraemer algorithm [31], whose goal is that of sampling in such a way that all legitimate class distributions are picked with equal probability. For each vector thus picked we randomly generate a test sample. We use this method for both the binary case and the multiclass case.

Note that this method is sharply different from traditional instantiations of the APP (as used, say, in [16, 32, 33, 19]), in which one

1. Chooses an integer P; this determines a "grid" g_1 of $(P+1)$ class prevalence values x/P, for $x \in \{0, ..., P\}$. For instance, given $P = 20$, this determines the grid $g_1 = \{0.00, 0.05, ..., 0.95, 1.00\}$ of 21 class prevalence values;
2. Generates the grid g_2 of the $K(P, n)$ probability distributions $(p_\sigma(y_1), ..., p_\sigma(y_n))$ such that all the class prevalence values $p_\sigma(y_i)$ are in g_1;
3. For each distribution p in the $K(P, n)$ probability distributions above, extracts m random samples of q data items each from U, in such a way that each extracted sample exhibits probability distribution p.
4. Use the extracted random samples for the evaluation of the quantifiers.

These traditional instantiations of the APP are suitable for small values of n, but quickly become unmanageable when n grows; for instance, in the binary case (n=2) we need to extract $m \cdot K(20, 2) = m \cdot 21$ samples, but this number grows to $m \cdot K(20, 3) = m \cdot 231$ for the ternary case, and quickly becomes unmanageable as n grows.[2]

[1] Everything we say here on how we generate the test samples also applies to how we generate the development samples.

[2] More precisely, there are $K(P, n) = \binom{P+n-1}{n-1}$ probability distributions $(p_\sigma(y_1), ..., p_\sigma(y_n))$ such that all the class prevalence values $p_\sigma(y_i)$ are in g_1. To exemplify, for $n = 5$ classes we already reach $K(20, 5) = 10,626$ valid combinations, while for $n = 10$ classes the number of combinations rises to $K(20, 10) = 10,015,005$.

3.3. The evaluation measures

In a recent theoretical study on the adequacy of evaluation measures for the quantification task [15], *relative absolute error* (RAE) and *absolute error* (AE) have been found to be the most satisfactory, and are thus the only measures used in LeQua 2022. In particular, as a measure we do not use the once widely used Kullback-Leibler Divergence (KLD), since the same study has found it to be unsuitable for evaluating quantification systems.[3] RAE and AE are defined as

$$\mathrm{RAE}(p_\sigma, \hat{p}_\sigma) = \frac{1}{n} \sum_{y \in \mathcal{Y}} \frac{|\hat{p}_\sigma(y) - p_\sigma(y)|}{p_\sigma(y)} \tag{1}$$

$$\mathrm{AE}(p_\sigma, \hat{p}_\sigma) = \frac{1}{n} \sum_{y \in \mathcal{Y}} |\hat{p}_\sigma(y) - p_\sigma(y)| \tag{2}$$

where p_σ is the true distribution on sample σ, $\hat{p}_\sigma$ is the predicted distribution, $\mathcal{Y}$ is the set of classes of interest, and $n = |\mathcal{Y}|$. Note that RAE is undefined when at least one of the classes $y \in \mathcal{Y}$ is such that its prevalence in the sample σ of unlabelled items is 0. To solve this problem, in computing RAE we smooth all $p_\sigma(y)$'s and $\hat{p}_\sigma(y)$'s via additive smoothing, i.e., we take $\underline{p}_\sigma(y) = (\epsilon + p_\sigma(y))/(\epsilon \cdot n + \sum_{y \in \mathcal{Y}} p_\sigma(y))$, where $\underline{p}_\sigma(y)$ denotes the smoothed version of $p_\sigma(y)$ and the denominator is just a normalising factor (same for the $\underline{\hat{p}}_\sigma(y)$'s); following Forman [4], we use the quantity $\epsilon = 1/(2|\sigma|)$ as the smoothing factor. In Equation 1 we then use the smoothed versions of $p_\sigma(y)$ and $\hat{p}_\sigma(y)$ in place of their original non-smoothed versions; as a result, RAE is now always defined.

As the official measure according to which systems are ranked, we use RAE; we also compute AE results, but we do not use them for ranking the systems. The official score obtained by a given quantifier is the average value of the official evaluation measure (RAE) across all test samples; for each system we also compute and report the value of AE. For each subtask in { T1A, T1B, T2A, T2B } we use a two-tailed t-test on related samples at different confidence levels ($\alpha = 0.05$ and $\alpha = 0.001$) to identify all participant runs that are *not* statistically significantly different from the best run, in terms of RAE and in terms of AE. We also compare all pairs of methods by means of *critical difference diagrams* (CD-diagrams – [34]). We adopt the Nemenyi test and set the confidence level to $\alpha = 0.05$. The test compares the average ranks in terms of RAE and takes into account the sample size $|\sigma|$.

3.4. Data

The data we have used are Amazon product reviews from a large crawl of such reviews. From the result of this crawl we have removed (a) all reviews shorter than 200 characters and (b) all reviews that have not been recognised as "useful" by any users; this has yielded the dataset Ω that we have used for our experimentation. As for the class labels, (i) for the two binary tasks (T1A and T2A) we have used two *sentiment* labels, i.e., `Positive` (which encompasses

[3] One reason why KLD is undesirable is that it penalizes differently underestimation and overestimation, and it does so opaquely, i.e., in a way that is not explicit from its mathematical form and that cannot be controlled via an explicit parameter; another is that it is very little robust to outliers. See [15, §4.7 and §5.2] for a detailed discussion of these and other reasons.

4-stars and 5-stars reviews) and `Negative` (which encompasses 1-star and 2-stars reviews), while for the two multiclass tasks (T1B and T2B) we have used 28 *topic* labels, representing the merchandise class the product belongs to (e.g., `Automotive`, `Baby`, `Beauty`).[4]

We have used the same data (training / development / test sets) for the binary vector task (T1A) and for the binary raw document task (T2A); i.e., the former are the vectorized (and shuffled) versions of the latter. Same for T1B and T2B. In order to generate the document vectors, we compute the average of the GloVe vectors [35] for the words contained in each document, thus producing 300-dimensional document embeddings. Each of the 300 dimensions of the document embeddings is then (independently) standardized, so that it has zero mean and unit variance.

The L_B (binary) training set and the L_M (multiclass) training set consist of 5,000 documents and 20,000 documents, respectively, sampled from the dataset Ω via *stratified sampling* so as to have "natural" prevalence values for all the class labels. (When doing stratified sampling for the binary "sentiment-based" task, we ignore the "topic" dimension; and when doing stratified sampling for the multiclass "topic-based" task, we ignore the "sentiment" dimension).

The development (validation) sets D_B (binary) and D_M (multiclass) consist of 1,000 development samples of 250 documents each (D_B) and 1,000 development samples of 1,000 documents each (D_M) generated from $\Omega \setminus L_B$ and $\Omega \setminus L_M$ via the Kraemer algorithm.

The test sets U_B and U_M consist of 5,000 test samples of 250 documents each (U_B) and 5,000 test samples of 1,000 documents each (U_M), generated from $\Omega \setminus (L_B \cup D_B)$ and $\Omega \setminus (L_M \cup D_M)$ via the Kraemer algorithm. A submission ("run") for a given subtask consists of prevalence estimations for the relevant classes (the two sentiment classes for the binary subtasks and the 28 topic classes for the multiclass subtasks) for each sample in the test set of that subtask.

3.5. Baselines

In order to set a sufficiently high bar for the participants to overcome, we made them aware of the availability of QuaPy [36], a library of quantification methods that contains, among others, implementations of a number of methods that have performed well in recent comparative evaluations.[5] QuaPy is a publicly available, open-source, Python-based framework that we have recently developed, and that implements not only learning methods, but also evaluation measures, parameter optimisation routines, and evaluation protocols, for LQ.

We used a number of quantification methods, as implemented in QuaPy, as baselines for the participants to overcome.[6] These methods were:

- **Maximum Likelihood Prevalence Estimation** (MLPE): Rather than a true quantification method, this a (more than) trivial baseline, consisting in assuming that the prevalence $p_\sigma(y_i)$ of a class y_i in the test sample σ is the same as the prevalence $p_L(y_i)$ that was observed for that class in the training set L.
- **Classify and Count** (CC): This is the trivial baseline, consisting in training a standard classifier h on the training set L, using it to classify all the data items $\mathbf{x}$ in the sample σ,

[4] The set of 28 topic classes is flat, i.e., there is no hierarchy defined upon it.

[5] https://github.com/HLT-ISTI/QuaPy

[6] Check the branch https://github.com/HLT-ISTI/QuaPy/tree/lequa2022

counting how many such items have been attributed to class y_i, doing this for all classes in $\mathcal{Y}$, and dividing the resulting counts by the cardinality $|\sigma|$ of the sample.

- **Probabilistic Classify and Count** (PCC) [3]: This is a probabilistic variant of CC where the "hard" classifier h is replaced with a "soft" (probabilistic) classifier s, and where counts are replaced with expected counts.
- **Adjusted Classify and Count** (ACC) [33]: This is an "adjusted" variant of CC in which the prevalence values predicted by CC are subsequently corrected by considering the misclassification rates of classifier h, as estimated on a held-out validation set. For our experiments, this held-out set consists of 40% of the training set.
- **Probabilistic Adjusted Classify and Count** (PACC) [3]: This is a probabilistic variant of ACC where the "hard" classifier h is replaced with a "soft" (probabilistic) classifier s, and where counts are replaced with expected counts. Equivalently, it is an "adjusted" variant of PCC in which the prevalence values predicted by PCC are corrected by considering the (probabilistic versions of the) misclassification rates of soft classifier s, as estimated on a held-out validation set. For our experiments, this held-out set consists of 40% of the training set.
- **HDy** [9]: This is a probabilistic binary quantification method that views quantification as the problem of minimising the divergence (measured in terms of the Hellinger Distance, HD) between two distributions of posterior probabilities returned by the classifier, one coming from the unlabelled examples and the other coming from a validation set consisting of 40% of the training documents. HDy seeks for the mixture parameter $\alpha \in [0, 1]$ that minimizes the HD between (a) the mixture distribution of posteriors from the positive class (weighted by α) and from the negative class (weighted by $(1 - \alpha)$), and (b) the unlabelled distribution.
- The **Saerens-Latinne-Decaestecker** algorithm (SLD) [32] (see also [37]): This is a method based on Expectation Maximization, whereby the posterior probabilities returned by a soft classifier s for data items in an unlabelled set U, and the class prevalence values for U, are iteratively updated in a mutually recursive fashion. For SLD we calibrate the classifier since, for reasons discussed in [37], this yields an advantage for this method. [7]
- **QuaNet** [16]: This is a deep learning architecture for quantification that predicts class prevalence values by taking as input (i) the class prevalence values as estimated by CC, ACC, PCC, PACC, SLD; (ii) the posterior probabilities $\Pr(y|\mathbf{x})$ for the positive class (since QuaNet is a binary method) for each document $\mathbf{x}$, and (iii) embedded representations of the documents. For task T1A, we directly use the vectorial representations that we have provided to the participants as the document embeddings, while for task T2A we use the RoBERTa embeddings (described below). For training QuaNet, we use the training set L for training the classifier. We then use the validation set for training the network parameters, using 10% of the validation samples for monitoring the validation loss (we apply early stop after 10 epochs that have shown no improvement). Since we devote the validation set to train part of the model, we did not carry out model selection for QuaNet, which was used with default hyperparameters (a learning rate of $1e^{-4}$, 64 dimensions in

[7] Calibration does not yield similar improvements for other methods such as PCC, PACC, and QuaNet, though. For this reason, we only calibrate the classifier for SLD.

the LSTM hidden layer, and a drop-out probability of 0.5).

All the above methods (with the exception of MLPE) are described in more detail in [19, §3.3 and §3.4], to which we refer the interested reader; all these methods are well-established, the most recent one (QuaNet) having been published in 2018. For all methods, we have trained the underlying classifiers via logistic regression, as implemented in the `scikit-learn` framework (https://scikit-learn.org/stable/index.html). Note that we have used HDy and QuaNet as baselines only in T1A and T2A, since they are binary-only methods. All other methods are natively multiclass, so we have used them in all four subtasks.

We optimize two hyperparameters of the logistic regression learner by exploring C (the inverse of the regularization strength) in the range $\{10^{-3}, 10^{-2}, \ldots, 10^{+3}\}$ and CLASS_WEIGHT (indicating the relative importance of each class) in {"balanced", "not-balanced"}. For each quantification method, model selection is carried out by choosing the combination of hyperparameters yielding the lowest average RAE across all validation samples.

For the raw documents subtasks (T2A and T2B), for each baseline quantification method we have actually generated *two* quantifiers, using two different methods for turning documents into vectors. (The only two baseline methods for which we do not do this are MLPE, which does not use vectors, and QuaNet, that internally generates its own vectors.) The two methods are

- The standard tfidf term weighting method, expressed as

$$\text{tfidf}(f, \mathbf{x}) = \log \#(f, \mathbf{x}) \times \log \frac{|L|}{|\mathbf{x}' \in L : \#(f, \mathbf{x}') > 0|} \tag{3}$$

  where $\#(f, \mathbf{x})$ is the raw number of occurrences of term f in document $\mathbf{x}$; weights are then normalized via cosine normalization, as

$$w(f, \mathbf{x}) = \frac{\text{tfidf}(f, \mathbf{x})}{\sqrt{\sum_{f' \in F} \text{tfidf}(f', \mathbf{x})^2}} \tag{4}$$

  where F is the set of all unigrams and bigrams that occur at least 5 times in L.
- The RoBERTa transformer [38], from the Hugging Face hub.[8] In order to use RoBERTa, we truncate the documents to the first 256 tokens, and fine-tune RoBERTa for the task of classification via prompt learning for a maximum of 10 epochs on our training data, thus taking the model parameters from the epoch which yields the best macro F_1 as monitored on a held-out validation set consisting of 10% of the training documents randomly sampled in a stratified way. For training, we set the learning rate to $1e^{-5}$, the weight decay to 0.01, and the batch size to 16, leaving the other hyperparameters at their default values. For each document, we generate features by first applying a forward pass over the fine-tuned network, and then averaging the embeddings produced for the special token [CLS] across all the 12 layers of RoBERTa. (In experiments that we carried out for another project, this latter approach yielded slightly better results than using the [CLS] embedding of the last layer alone.) The embedding size of RoBERTa, and hence the number of dimensions of our vectors, amounts to 768.

[8] https://huggingface.co/docs/transformers/model_doc/roberta

Table 1
The teams who participated in LeQua 2022 and the tasks for which they submitted runs.

	T1A	T1B	T2A	T2B
DortmundAI	x	x		
KULeuven	x	x		
UniLeiden	x			
UniOviedo(Team1)	x	x	x	x
UniOviedo(Team2)	x	x		
UniPadova			x	

4. The participating systems

Six teams submitted runs to LeQua 2022. As shown in in Table 1, the most popular subtask was, unsurprisingly, T1A (5 teams), while the subtask with the smallest participation was T2B (1 team). We here list the teams in alphabetical order:

- **DortmundAI** [39] submitted a run each for T1A and T1B. Their original goal was to use a modified version of the SLD algorithm described in Section 3.5. The modification introduced by DortmundAI consists of the use of a regularization technique meant to smooth the estimates that expectation maximization computes for the class prevalence values at each iteration. After extensively applying model selection, though, the team realized that the best configurations of hyperparameters often reduce the strength of such regularization, so as to make the runs produced by their regularized version of SLD almost identical to a version produced by using the "traditional" SLD algorithm. They also found that a thorough optimization of the hyperparameters of the base classifier was instead the key to producing good results.
- **KULeuven** [40] submitted a run each for T1A and T1B. Their system consisted of a robust calibration of the SLD [32] method based on the observations of Molinari et al. [41]. While the authors explored *trainable calibration strategies* (i.e., regularization constraints that modify the training objective of a classifier in favour of better calibrated solutions), the team finally contributed a solution based on the Platt rescaling [42] of the SVM outputs (i.e., a *post-hoc calibration method* that is applied after training the classifier) which they found to perform better in validation. Their solution differs from the version of SLD provided as baseline mainly in the choice of the underlying classifier (the authors chose SVMs while the provided baseline is based on logistic regression) and in the amount of effort devoted to the optimization of the hyperparameters (which was higher in the authors' case).
- **UniLeiden** [43] submitted a run for T1A only. The authors' system is a variant of the Median Sweep (MS) method proposed by Forman [4, 44], called *Simplified Continuous Sweep*, which consists of a smooth adaptation of the original method. The main modifications come down to computing the mean (instead of the median) of the class prevalence

estimates by integrating over continuous functions (instead of summing across discrete functions) that represent the classification counts and misclassification rates. Since the underlying distributions of these counts and rates are unknown, kernel density estimation is used to approximate them. Although the system did not yield improved results with respect to MS, it paves the way for better understanding the theoretical implications of MS.

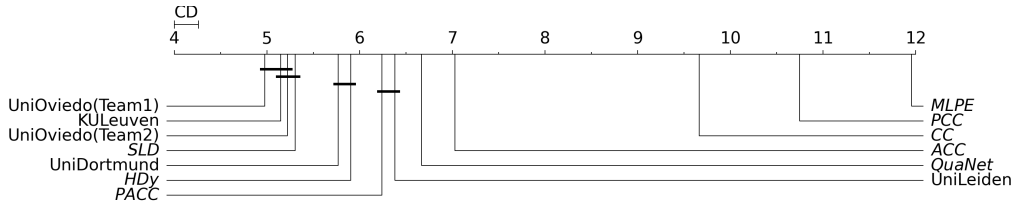
- **UniOviedo(Team1)** [45] submitted a run each for all four subtasks. Their system consists of a deep neural network architecture explicitly devised for the quantification task. The learning method is non-aggregative and does not need to know the labels of the training items composing a sample. As the training examples to train the quantifiers that produced the submissions it used the samples with known prevalence from the development sets D_B and D_M (each set is used for its respective task). A generator of additional samples that produces mixtures of pairs of samples of known prevalence is used to increase the number of training examples. Data from training sets L_B and L_M are used only to generate additional training samples when over-fitting is observed. Every sample is represented as a set of histograms, each one representing the distribution of values of an input feature. For tasks T1A and T1B, histograms are directly computed on the input vectors. For tasks T2A and T2B, the input text are first converted into dense vectors using a BERT model, for which the histograms are computed. The network uses RAE as the loss function, modified by the smoothing parameter so as to avoid undefined values when a true prevalence is zero, thus directly optimizing the official evaluation measure.
- **UniOviedo(Team2)** [46] submitted a run each for T1A and T1B. For T1A, this team used a highly optimized version of the HDy system (that was also one of the baseline systems), obtained by optimizing three different parameters (similarity measure used, number of bins used, method used for binning the posteriors returned by the classifier). For T1B, this team used a version of HDy (called EDy) different from the previous one; EDy uses, for the purpose of measuring the distance between two histograms, the "energy distance" in place of the Hellinger Distance.
- **UniPadova** [47] submitted a run for T2A only. Their system consisted of a classify-and-count method in which the underlying classifier is a probabilistic "BM25" classifier. The power of this method thus only derives from the term weighting component, since nothing in the method makes explicit provisions for distribution shift.

5. Results

In this section we discuss the results obtained by our participant teams in the four subtasks we have proposed. The evaluation campaign started on Dec 1, 2021, with the release of the training sets (L_B and L_M) and of the development sets (D_B and D_M); alongside them, the participant teams were provided with a dummy submission, a format checker, and the official evaluation script. The unlabelled test sets (U_B and U_M) were released on Apr 22, 2022; and runs had to be submitted by May 11, 2022. Each team could submit up to two runs per subtask, provided each such run used a truly different method (and not, say, the same method using different parameter values); however, no team decided to take advantage of this, and each team submitted at most

Rank	Run	RAE	AE
1	KULeuven	**0.10858** ± 0.27476	0.02418 ± 0.01902
2	UniOviedo(Team1)	0.10897‡ ± 0.21887	**0.02327** ± 0.01811
3	UniOviedo(Team2)	0.11130‡ ± 0.23144	0.02475 ± 0.01908
4	SLD	0.11382‡ ± 0.26605	0.02518 ± 0.01977
5	UniDortmund	0.11403† ± 0.20345	0.02706 ± 0.02096
6	HDy	0.14514 ± 0.45617	0.02814 ± 0.02212
7	$PACC$	0.15218 ± 0.46435	0.02985 ± 0.02258
8	ACC	0.17020 ± 0.50795	0.03716 ± 0.02935
9	UniLeiden	0.19624 ± 0.82620	0.03171 ± 0.02424
10	$QuaNet$	0.31764 ± 1.35223	0.03418 ± 0.02527
11	CC	1.08400 ± 4.31046	0.09160 ± 0.05539
12	PCC	1.39402 ± 5.62067	0.11664 ± 0.06977
13	$MLPE$	3.26692 ± 14.85223	0.32253 ± 0.22961

(a)



(b)

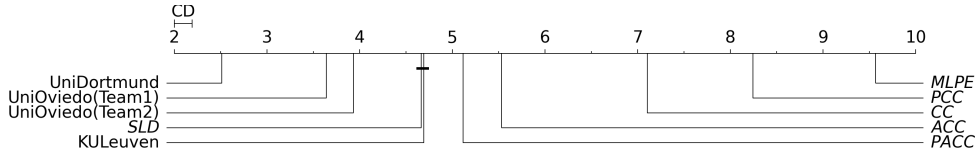
Figure 1: Results of Task T1A. Table (a) reports the results of participant teams in terms of RAE (official measure for ranking) and AE, averaged across the 5,000 test samples. **Boldface** indicates the best method for a given evaluation measure. Superscripts † and ‡ denote the methods (if any) whose scores are *not* statistically significantly different from the best one according to a paired sample, two-tailed t-test at different confidence levels: symbol † indicates $0.001 < p$-value < 0.05 while symbol ‡ indicates $0.05 \leq p$-value. The absence of any such symbol indicates p-value ≤ 0.001 (i.e., that the difference in performance between the method and the best one is statistically significant at a high confidence level). Baseline methods are typeset in *italic*. Subfigure (b) reports the CD-diagram for Task T1A for the averaged ranks in terms of RAE.

one run per subtask. An instantiation of Codalab (https://codalab.org/) was set up in order to allow the teams to submit their runs. The true labels of the unlabelled test sets were released on May 13, 2022, after the submission period was over and the official results had been announced to the participants. In the rest of this section we discuss the results that the participants' systems and the baseline systems have obtained in the vector subtasks (T1A and T1B – Section 5.1), in the raw document subtasks (T2A and T2B – Section 5.2), in the binary subtasks (T1A and T2A – Section 5.3), and in the multiclass subtasks (T1B and T2B – Section 5.4).

We report the results of the participants' systems and the baseline systems in Figure 1 (for subtask T1A), Figure 2 (T1B), Figure 3 (T2A), and Figure 4 (T2B). In each such figure we also display critical-distance diagrams illustrating how the systems rank in terms of RAE and when the difference between the systems is statistically significant.

Rank	Run	RAE	AE
1	UniDortmund	**0.87987** ± 0.75139	**0.01173** ± 0.00284
2	UniOviedo(Team1)	0.88415‡ ± 0.45537	0.02799 ± 0.00723
3	UniOviedo(Team2)	1.11395 ± 0.92516	0.01178‡ ± 0.00329
4	KULeuven	1.17798 ± 1.05501	0.01988 ± 0.00395
5	SLD	1.18207 ± 1.09757	0.01976 ± 0.00399
6	$PACC$	1.30538 ± 0.98827	0.01578 ± 0.00379
7	ACC	1.42134 ± 1.26958	0.01841 ± 0.00437
8	CC	1.89365 ± 1.18721	0.01406 ± 0.00295
9	PCC	2.26462 ± 1.41613	0.01711 ± 0.00332
10	$MLPE$	4.57675 ± 4.51384	0.04227 ± 0.00414

(a)



(b)

Figure 2: As in Figure 1, but for T1B in place of T1A.

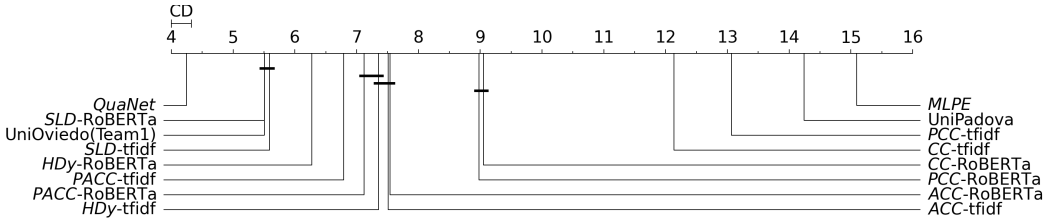
Interestingly enough, no system (either participants' system or baseline system) was the best performer in more than one subtask, with four different systems (the **KULeuven** system for T1A, the **DortmundAI** system for T1B, the **QuaNet** baseline system for T2A, and the **UniOviedo(Team1)** system for T2B) claiming top spot for the four subtasks. Overall, the performance of UniOviedo(Team1) was especially noteworthy since, aside from topping the rank in T2B, it obtained results not statistically significantly different ($0.05 \leq p$-value) from those of the top-performing team also in T1A and T1B.

The results allow us to make a number of observations. We organize the discussion of these results in four sections (Section 5.1 to Section 5.4), one for each of the four dimensions (vectors vs. raw documents, binary vs. multiclass) according to which the four subtasks are structured. However, before doing that, we discuss some conclusions that may be drawn from the results and that affect all four dimensions.

1. MLPE is the worst predictor. This is true in all four subtasks, and was expected, given the fact that the test data are generated by means of the APP, which implies that the test data contain a very high number of samples characterized by substantial distribution shift, and that on these samples MLPE obviously performs badly.
2. CC and PCC obtain very low quantification accuracy; this is the case in all four subtasks, where these two methods are always near the bottom of the ranking. This confirms the fact (already recorded in previous work – see e.g., [36, 19, 48]) that they are not good performers when the APP is used for generating the dataset, i.e., they are not good performers when there is substantial distribution shift. Interestingly enough, CC always outperforms PCC, which was somehow unexpected.

Rank	Run	RAE	AE
1	$QuaNet$	**0.07805** ± 0.25437	**0.01306** ± 0.01009
2	SLD-tfidf	0.08703† ± 0.16721	0.01952 ± 0.01543
3	UniOviedo(Team1)	0.10704 ± 0.27896	0.01916 ± 0.01467
4	HDy-tfidf	0.12198 ± 0.17207	0.02914 ± 0.02266
5	SLD-RoBERTa	0.13616 ± 0.45312	0.02208 ± 0.01562
6	$PACC$-tfidf	0.13804 ± 0.48977	0.02626 ± 0.02080
7	ACC-tfidf	0.16113 ± 0.54750	0.03090 ± 0.02443
8	HDy-RoBERTa	0.16285 ± 0.55900	0.02421 ± 0.01612
9	$PACC$-RoBERTa	0.32902 ± 1.46314	0.03227 ± 0.02381
10	ACC-RoBERTa	0.33023 ± 1.49746	0.03374 ± 0.02539
11	CC-RoBERTa	0.41222 ± 1.81806	0.04053 ± 0.02976
12	PCC-RoBERTa	0.45182 ± 1.92703	0.04077 ± 0.02817
13	CC-tfidf	1.06748 ± 4.83335	0.10286 ± 0.07348
14	PCC-tfidf	1.36165 ± 6.37488	0.14414 ± 0.10237
15	UniPadova	3.02245 ± 11.99428	0.25067 ± 0.14675
16	$MLPE$	3.26692 ± 14.85223	0.32253 ± 0.22961

(a)



(b)

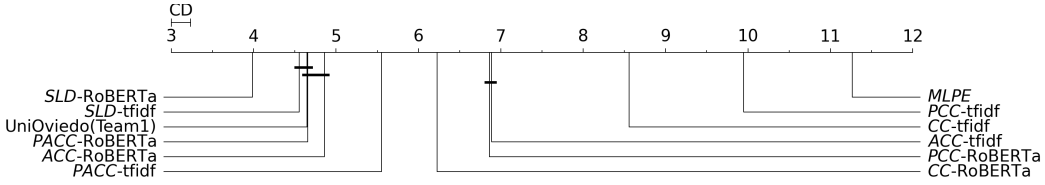
Figure 3: As in Figure 1, but for T2A in place of T1A.

3. ACC and PACC are mid-level performers; this holds in all four subtasks, in which both methods are always in the middle portion of the ranking. Interestingly enough, PACC always outperforms ACC, somehow contradicting the impression (see Bullet 2) that "hard" counts are better than expected counts and/or that the calibration routine has not done a good job.
4. SLD is the strongest baseline; this is true in all four subtasks, in which SLD, while never being the best performer, is always in the top ranks. This confirms the fact (already recorded in previous work – see e.g., [36, 19, 48]) that SLD is a very strong performer when the APP is used for generating the dataset, i.e., when the test data contain many samples characterized by substantial distribution shift.
5. Overall, the ranking MLPE $<$ PCC $<$ CC $<$ ACC $<$ PACC $<$ SLD (where "$<$" means "performs worse than") clearly emerges from all four tasks.

As it might be expected, not always a good performance according to RAE (our official measure) also corresponds to a good performance on AE (our other measure). Only in 2 subtasks out of 4

Rank	Run	RAE	AE
1	UniOviedo(Team1)	**1.23085** ± 0.72831	0.03208 ± 0.00921
2	SLD-RoBERTa	1.30978 ± 1.61205	0.01552 ± 0.00439
3	SLD-tfidf	1.31950 ± 1.23382	0.01829 ± 0.00376
4	$PACC$-RoBERTa	1.45429 ± 1.00967	**0.01220** ± 0.00260
5	ACC-RoBERTa	1.48661 ± 1.07152	0.01310 ± 0.00290
6	$PACC$-tfidf	1.53853 ± 1.43093	0.01789 ± 0.00508
7	CC-RoBERTa	1.69071 ± 1.15729	0.01367 ± 0.00296
8	PCC-RoBERTa	1.77143 ± 1.15163	0.01328 ± 0.00272
9	ACC-tfidf	2.01440 ± 2.16362	0.01993 ± 0.00548
10	CC-tfidf	2.24393 ± 1.52031	0.01949 ± 0.00399
11	PCC-tfidf	3.06004 ± 2.21288	0.02913 ± 0.00469
12	$MLPE$	4.57675 ± 4.51384	0.04227 ± 0.00414

(a)



(b)

Figure 4: As in Figure 1, but for T2B in place of T1A.

(T1B, with the DortmundAI system, and T2A, with the QuaNet baseline system) the system that scores best according to RAE also scores best according to AE; in the other 2 subtasks this is not the case, and in one case (T2B) the system that performs best according to RAE (the UniOviedo(Team1) system) has a very low performance according to AE. This suggests that for some systems, including the UniOviedo(Team1) system, parameter optimization (which, quite naturally, is performed by trying to optimize the official measure) may have played an especially important role.

5.1. T1A and T1B: The vector subtasks

In the vector subtasks the top-performing systems, KULeuven for T1A and UniDortmund for T1B, both consist of carefully optimized instances of SLD. The KULeuven system outperformed all the baseline systems in both tasks, while the UniDortmund system ranked 5th in T1A, one position below the SLD baseline.

The runs from UniOviedo(Team1) and UniOviedo(Team2) obtained 2nd and 3rd ranks, respectively, in both T1A and T1B. The UniOviedo(Team1) system performed very well in both cases, obtaining RAE scores that, according to the test of statistical significance, are not significantly different from the best result obtained in each of these subtasks. Things are different if we instead look at the AE scores, for which UniOviedo(Team1) obtained the best result in T1A but the second-worst result in T1B.

5.2. T2A and T2B: The raw documents subtasks

In both raw document tasks (T2A and T2B) the best-performing methods is always one based on deep learning (the QuaNet baseline for T2A and the UniOviedo(Team1) system for T2B).

A direct comparison between the UniOviedo(Team1) system and QuaNet in the multiclass case (T2B) is not possible because QuaNet is a binary-only method (see Section 3.5) and was thus not used in T2B. A common characteristic between these two methods is that both use (part of the) samples from the validation data not for tuning hyperparameters but for training the model.

Concerning the baseline systems, the results do not give a definitive answer on which between tfidf and RoBERTa is the best method for mapping raw documents into vectors. In fact, out of 9 cases (5 for T2A, 4 for T2B) in which we have generated both variants of the same baseline, the tfidf variant outperforms the RoBERTa variant in 4 cases and is outperformed by it in 5 cases. This was unexpected, since RoBERTa is a way more sophisticated and modern method than the time-worn tfidf. Interestingly (and mysteriously) enough, the tfidf variant is almost always the better performer in the binary case (T2A – 4 cases out of 5), while the RoBERTa variant always outperforms the tfidf variant in the multiclass case (T2B – 4 cases out of 4).

5.3. T1A and T2A: The binary subtasks

Concerning T1A and T2A (the binary subtasks), we should first observe that we here use two further baseline systems, namely, HDy and QuaNet; we only use them in the binary subtasks since they are not natively multiclass. HDy performs fairly well in both T1A and T2A, outperforming MLPE, PCC, CC, ACC, and PACC (but not SLD) in both cases. Instead, QuaNet performs less consistently, since it places in the mid-lower ranks of the table in T1A but is no less than the best performer in T2A.

The inconsistent results obtained by QuaNet on binary tasks contrast with those obtained by the UniOviedo(Team1) system, the other method based on deep learning, which ranks among the top positions in both T1A and T2A. This is somehow surprising, given that in T1A (unlike in T2A), the source vectors used by UniOviedo(Team1) and QuaNet methods were exactly the same.

5.4. T1B and T2B: The multiclass subtasks

Regarding the multiclass subtasks, the UniOviedo(Team1) system stands out, since it consistently obtained results that either outperform all other methods (T2B) or were not different, in a statistically significant sense, from the best-performing method (T1B). UniOviedo(Team1) was the only team participating in the raw-document multiclass subtask T2B. Although UniOviedo(Team1) beat all other baselines in terms of RAE, it performed comparably worse in terms of AE to most of the baselines (actually, worse than all baselines but MLPE).

6. Final remarks

Overall, something that we learn from this shared task is that SLD is very hard to beat (thereby confirming recent results reported in [19, 36, 48]), and that it tends to fare very well across

different settings, including binary and multiclass quantification problems, and including different ways of processing text. This observation is reinforced by the fact that two of the best-performing systems (KULeuven and UniDortmund, which placed 1st in T1A and T1B, respectively) actually consist of carefully-tuned instances of SLD. Another "classic" method that has also proven to behave well is HDy, a method that forms the basis on which one of the best-performing methods (UniOviedo(Team2)) is built upon. However, the system that has delivered the most consistently competitive results across all tasks (UniOviedo(Team1)) is a "non-classical" one, since it is based on deep-learning technology.

To conclude, we think that LeQua 2022 has proven very useful for the quantification community, since it has confirmed, in a controlled settings, some intuitions about "classic" quantification systems (e.g., SLD) that had already surfaced in the recent literature, but has also shown that there are margins of improvement over them, especially if using "deep" learning approaches (such as QuaNet and the system used by UniOviedo(Team1)).

We plan to propose a LeQua edition for CLEF 2023, so as to allow the LeQua 2022 participants to profit from their 2022 experience in order to consolidate their systems so as to improve on their 2022 performance, and so as to allow prospective participants who could not make it for 2022 to jump in. The experimental setting that we have used for LeQua 2022 will be the starting point, but we might want to incorporate in it possible suggestions that might arise during the LeQua session at the CLEF 2022 conference.

This session will host (a) a keynote talk by George Forman (Amazon Research), (b) a detailed presentation by the organisers, overviewing the lab and the results of the participants, (c) oral presentations by the participating teams, and (d) a final discussion on the takeaway message that LeQua 2022 gives us.

Acknowledgments

This work has been supported by the SoBigData++ project, funded by the European Commission (Grant 871042) under the H2020 Programme INFRAIA-2019-1, and by the AI4Media project, funded by the European Commission (Grant 951911) under the H2020 Programme ICT-48-2020. The authors' opinions do not necessarily reflect those of the European Commission. We thank Alberto Barron Cedeño, Juan José del Coz, Preslav Nakov, and Paolo Rosso, for advice on how to best set up this lab.

References

[1] J. J. del Coz, P. González, A. Moreo, F. Sebastiani, Learning to quantify: Methods and applications (LQ 2021), in: Proceedings of the 30th ACM International Conference on Knowledge Management (CIKM 2021), Gold Coast, AU, 2021, pp. 4874–4875. doi:10.1145/3459637.3482040.

[2] P. González, A. Castaño, N. V. Chawla, J. J. del Coz, A review on quantification learning, ACM Computing Surveys 50 (2017) 74:1–74:40. doi:10.1145/3117807.

[3] A. Bella, C. Ferri, J. Hernández-Orallo, M. J. Ramírez-Quintana, Quantification via prob-

ability estimators, in: Proceedings of the 11th IEEE International Conference on Data Mining (ICDM 2010), Sydney, AU, 2010, pp. 737–742. doi:10.1109/icdm.2010.75.

[4] G. Forman, Quantifying counts and costs via classification, Data Mining and Knowledge Discovery 17 (2008) 164–206. doi:10.1007/s10618-008-0097-y.

[5] R. Levin, H. Roitman, Enhanced probabilistic classify and count methods for multi-label text quantification, in: Proceedings of the 7th ACM International Conference on the Theory of Information Retrieval (ICTIR 2017), Amsterdam, NL, 2017, pp. 229–232. doi:10.1145/3121050.3121083.

[6] J. Barranquero, J. Díez, J. J. del Coz, Quantification-oriented learning based on reliable classifiers, Pattern Recognition 48 (2015) 591–604. doi:10.1016/j.patcog.2014.07.032.

[7] G. Da San Martino, W. Gao, F. Sebastiani, Ordinal text quantification, in: Proceedings of the 39th ACM Conference on Research and Development in Information Retrieval (SIGIR 2016), Pisa, IT, 2016, pp. 937–940. doi:10.1145/2911451.2914749.

[8] A. Esuli, F. Sebastiani, Optimizing text quantifiers for multivariate loss functions, ACM Transactions on Knowledge Discovery and Data 9 (2015) Article 27. doi:10.1145/2700406.

[9] V. González-Castro, R. Alaiz-Rodríguez, E. Alegre, Class distribution estimation based on the Hellinger distance, Information Sciences 218 (2013) 146–164. doi:10.1016/j.ins.2012.05.028.

[10] L. Milli, A. Monreale, G. Rossetti, F. Giannotti, D. Pedreschi, F. Sebastiani, Quantification trees, in: Proceedings of the 13th IEEE International Conference on Data Mining (ICDM 2013), Dallas, US, 2013, pp. 528–536. doi:10.1109/icdm.2013.122.

[11] V. Vapnik, Statistical learning theory, Wiley, New York, US, 1998.

[12] J. G. Moreno-Torres, T. Raeder, R. Alaíz-Rodríguez, N. V. Chawla, F. Herrera, A unifying view on dataset shift in classification, Pattern Recognition 45 (2012) 521–530. doi:10.1016/j.patcog.2011.06.019.

[13] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, N. D. Lawrence (Eds.), Dataset shift in machine learning, The MIT Press, Cambridge, US, 2009. doi:10.7551/mitpress/9780262170055.001.0001.

[14] A. Esuli, F. Sebastiani, Sentiment quantification, IEEE Intelligent Systems 25 (2010) 72–75.

[15] F. Sebastiani, Evaluation measures for quantification: An axiomatic approach, Information Retrieval Journal 23 (2020) 255–288. doi:10.1007/s10791-019-09363-y.

[16] A. Esuli, A. Moreo, F. Sebastiani, A recurrent neural network for sentiment quantification, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM 2018), Torino, IT, 2018, pp. 1775–1778. doi:10.1145/3269206.3269287.

[17] W. Gao, F. Sebastiani, From classification to quantification in tweet sentiment analysis, Social Network Analysis and Mining 6 (2016) 1–22. doi:10.1007/s13278-016-0327-z.

[18] A. Esuli, A. Moreo, F. Sebastiani, Cross-lingual sentiment quantification, IEEE Intelligent Systems 35 (2020) 106–114. doi:10.1109/MIS.2020.2979203.

[19] A. Moreo, F. Sebastiani, Tweet sentiment quantification: An experimental re-evaluation, PLoS ONE (2022). Forthcoming.

[20] L. Milli, A. Monreale, G. Rossetti, D. Pedreschi, F. Giannotti, F. Sebastiani, Quantification

in social networks, in: Proceedings of the 2nd IEEE International Conference on Data Science and Advanced Analytics (DSAA 2015), Paris, FR, 2015. doi:10.1109/dsaa.2015.7344845.

[21] A. G. Maletzke, D. Moreira dos Reis, G. E. Batista, Combining instance selection and self-training to improve data stream quantification, Journal of the Brazilian Computer Society 24 (2018) 43–48. doi:10.1186/s13173-018-0076-0.

[22] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, V. 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, pp. 1–18. doi:10.18653/v1/s16-1001.

[23] R. Higashinaka, K. Funakoshi, M. Inaba, Y. Tsunomori, T. Takahashi, N. Kaji, Overview of the 3rd Dialogue Breakdown Detection challenge, in: Proceedings of the 6th Dialog System Technology Challenge, Long Beach, US, 2017.

[24] Z. Zeng, S. Kato, T. Sakai, Overview of the NTCIR-14 Short Text Conversation task: Dialogue Quality and Nugget Detection subtasks, in: Proceedings of the 14th Workshop on NII Testbeds and Community for Information access Research (NTCIR 2019), Tokyo, JP, 2019, pp. 289–315.

[25] Z. Zeng, S. Kato, T. Sakai, I. Kang, Overview of the NTCIR-15 Dialogue Evaluation task (DialEval-1), in: Proceedings of the 15th Workshop on NII Testbeds and Community for Information access Research (NTCIR 2020), Tokyo, JP, 2020, pp. 13–34.

[26] R. Alaíz-Rodríguez, A. Guerrero-Curieses, J. Cid-Sueiro, Class and subclass probability re-estimation to adapt a classifier in the presence of concept drift, Neurocomputing 74 (2011) 2614–2623. doi:10.1016/j.neucom.2011.03.019.

[27] M. C. du Plessis, G. Niu, M. Sugiyama, Class-prior estimation for learning from positive and unlabeled data, Machine Learning 106 (2017) 463–492. doi:10.1007/s10994-016-5604-6.

[28] G. King, Y. Lu, Verbal autopsy methods with multiple causes of death, Statistical Science 23 (2008) 78–91. doi:10.1214/07-sts247.

[29] D. Card, N. A. Smith, The importance of calibration for estimating proportions from annotations, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2018), New Orleans, US, 2018, pp. 1636–1646. doi:10.18653/v1/n18-1148.

[30] D. J. Hopkins, G. King, A method of automated nonparametric content analysis for social science, American Journal of Political Science 54 (2010) 229–247. doi:10.1111/j.1540-5907.2009.00428.x.

[31] N. A. Smith, R. W. Tromble, Sampling uniformly from the unit simplex, Technical Report, Johns Hopkins University, 2004. https://www.cs.cmu.edu/~nasmith/papers/smith+tromble.tr04.pdf.

[32] M. Saerens, P. Latinne, C. Decaestecker, Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure, Neural Computation 14 (2002) 21–41. doi:10.1162/089976602753284446.

[33] G. Forman, Counting positives accurately despite inaccurate classification, in: Proceedings of the 16th European Conference on Machine Learning (ECML 2005), Porto, PT, 2005, pp. 564–575. doi:10.1007/11564096_55.

[34] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine

Learning Research 7 (2006) 1–30.
[35] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 12th Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), Doha, QA, 2014, pp. 1532–1543.
[36] A. Moreo, A. Esuli, F. Sebastiani, QuaPy: A Python-based framework for quantification, in: Proceedings of the 30th ACM International Conference on Knowledge Management (CIKM 2021), Gold Coast, AU, 2021, pp. 4534–4543. doi:10.1145/3459637.3482015.
[37] A. Esuli, A. Molinari, F. Sebastiani, A critical reassessment of the Saerens-Latinne-Decaestecker algorithm for posterior probability adjustment, ACM Transactions on Information Systems 39 (2021) Article 19. doi:10.1145/3433164.
[38] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, RoBERTa: A robustly optimized BERT pretraining approach, 2019. ArXiv:1907.11692.
[39] M. Senz, M. Bunse, DortmundAI at LeQua 2022: Regularized SLD, in: Working Notes of the 2022 Conference and Labs of the Evaluation Forum (CLEF 2022), Bologna, IT, 2022.
[40] T. Popordanoska, M. B. Blaschko, KULeuven at LeQua 2022: Model calibration in quantification learning, in: Working Notes of the 2022 Conference and Labs of the Evaluation Forum (CLEF 2022), Bologna, IT, 2022.
[41] A. Molinari, A. Esuli, F. Sebastiani, Active learning and the Saerens-Latinne-Decaestecker algorithm: An evaluation, in: Proceedings of the 2nd Joint Conference of the Information Retrieval Communities in Europe (CIRCLE 2022), Samatan, FR, 2022.
[42] J. C. Platt, Probabilistic outputs for support vector machines and comparison to regularized likelihood methods, in: A. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans (Eds.), Advances in Large Margin Classifiers, The MIT Press, Cambridge, MA, 2000, pp. 61–74.
[43] K. Kloos, Q. A. Meertens, J. D. Karch, UniLeiden at LeQua 2022: The first step in understanding the behaviour of the median sweep quantifier using continuous sweep, in: Working Notes of the 2022 Conference and Labs of the Evaluation Forum (CLEF 2022), Bologna, IT, 2022.
[44] G. Forman, Quantifying trends accurately despite classifier error and class imbalance, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006), Philadelphia, US, 2006, pp. 157–166. doi:10.1145/1150402.1150423.
[45] P. González, UniOviedo(Team1) at LeQua 2022: Sample-based quantification using deep learning, in: Working Notes of the 2022 Conference and Labs of the Evaluation Forum (CLEF 2022), Bologna, IT, 2022.
[46] J. J. del Coz, UniOviedo(Team2) at LeQua 2022: Comparison of traditional quantifiers and a new method based on energy distance, in: Working Notes of the 2022 Conference and Labs of the Evaluation Forum (CLEF 2022), Bologna, IT, 2022.
[47] G. M. Di Nunzio, UniPadova at LeQua 2022: A preliminary study of a Tidyverse approach to quantification, in: Working Notes of the 2022 Conference and Labs of the Evaluation Forum (CLEF 2022), Bologna, IT, 2022.
[48] A. Moreo, F. Sebastiani, Re-assessing the "classify and count" quantification method, in: Proceedings of the 43rd European Conference on Information Retrieval (ECIR 2021), volume II, Lucca, IT, 2021, pp. 75–91.