UniOviedo(Team2) at LeQua 2022: Comparison of traditional quantifiers and a new method based on Energy Distance

Juan José del Coz[1]

[1]*Artificial Intelligence Center at Gijón, University of Oviedo, Spain*

Abstract

The idea of this team was to compare the performance of some of the most important quantification methods and a new approach based on the Energy Distance that has been proposed by our group recently. This paper describes this method, called EDy, and the experimentation carried out to tackle the vector subtasks (*T1A* and *T1B*) of LeQua 2022 quantification competition.

Keywords

LaTeX quantification, prevalence estimation, energy distance

1. Motivation

Our main intention in this competition was to analyze the behavior of a new quantification algorithm devised by our group. This method, called EDy, is unpublished yet and will be briefly described in Section 2.2. To assess its performance, we compare it with some of the most popular quantification algorithms, see Section 2.1.

We just focus in vector subtasks (*T1A* and *T1B*) because we are not experts on deep learning that is more or less required to tackle the subtasks using raw documents (*T2A* and *T2B*). According to our previous studies using EDy over benchmark data, our hopes of being truly competitive were centered on *T1B*, because EDy usually provides better results for multiclass quantification tasks. In fact, we only submitted the scores of EDy for *T1B*. For the binary subtask *T1A* we employed HDy [1] with some customization. We achieved a broze medal in both competitions, but as we will see later, our results could easily have been better in subtask *T1B*.

2. Methods

Before describing the methods used, we introduce here some notation. In the general setting, quantification methods learn from a training set, $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, in which $\boldsymbol{x}_i$ is the description of an instance using the features of the input space, and y_i is its class. In the tasks of LeQua competition $y_i \in \{c_1, \ldots, c_k\}$ being $k = 2$ for binary tasks *TA* and $k = 28$ for multiclass

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy
juanjo@uniovi.es (J. J. del Coz)
www.aic.uniovi.es/juanjo (J. J. del Coz)
0000-0002-4288-3839 (J. J. del Coz)

quantification tasks *TB*. The goal of quantification learning is to automatically obtain models able to predict the prevalence of all classes, $\hat{\boldsymbol{p}} = \{\hat{p}_1, \ldots, \hat{p}_k\}$, given a set of unlabeled examples, $T = \{\boldsymbol{x}_j\}_{j=1}^{m}$, ensuring that $\hat{p}_l \geq 0$ and $\sum_{l=1}^{k} \hat{p}_l = 1$.

2.1. SOTA quantifiers

There are several quantification methods that can be considered state-of-the-art, see [2]. We chose the best performing methods according to our experience, namely:

- EM [3]. This method is based on expectation–maximization algorithm in which the parameters to be estimated are the class prior probabilities. This method is denoted as EMQ in QuaPy [4] and SLD in the baseline results given by the organizers[1].
- HDy [1]. This is a matching distribution quantifier that uses histograms to represent the distributions and the Hellinger Distance to measure histogram similarity.

However, while we used the EM method without any major customization, we tested two possible improvements for the HDy method:

1. A different way of computing the histograms. The original method is based on equal-width bins. We tested also equal-count bins, considering the examples of all the classes.
2. Taken into account the results reported in [5], we also tested Topsøe as similarity measure.

We improved the HDy results provided by the organizers using these modifications.

2.2. EDy

EDy is based on the method presented in [6]. It is also a matching distribution algorithm, like HDy, but the distributions are represented by the complete sets of examples (the sets with the training examples for each class, denoted as D^{c_l}, and the testing set T), and the metric is the Energy Distance (ED). Formally, EDy minimizes the ED between T and the weighted mixture distribution,

$$D' = D^{c_1} \cdot \hat{p}_1 + D^{c_2} \cdot \hat{p}_2 + \ldots + D^{c_k} \cdot \hat{p}_k. \tag{1}$$

with respect to $\hat{\boldsymbol{p}}$. That is:

$$\begin{aligned} \min_{\hat{p}_1,\ldots,\hat{p}_k} \quad & 2 \cdot \mathbb{E}_{\boldsymbol{x}_i \backsim D', \boldsymbol{x}_j \backsim T}\ \delta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ & -\mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}'_i \backsim D'}\ \delta(\boldsymbol{x}_i, \boldsymbol{x}'_i) - \mathbb{E}_{\boldsymbol{x}_j, \boldsymbol{x}'_j \backsim T}\ \delta(\boldsymbol{x}_j, \boldsymbol{x}'_j), \end{aligned} \tag{2}$$

where δ is a distance. The last term can be removed (it does not depend on $\hat{\boldsymbol{p}}$), so we have:

$$\begin{aligned} \min_{\hat{p}_1,\ldots,\hat{p}_k} \quad & 2 \sum_{l=1}^{k} \hat{p}_l\ \mathbb{E}_{\boldsymbol{x}_i \backsim D^{c_l}, \boldsymbol{x}_j \backsim T}\ \delta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ & - \sum_{l=1}^{k} \sum_{l'=1}^{k} \hat{p}_l\ \hat{p}_{l'}\ \mathbb{E}_{\boldsymbol{x}_i \backsim D^{c_l}, \boldsymbol{x}'_i \backsim D^{c_{l'}}}\ \delta(\boldsymbol{x}_i, \boldsymbol{x}'_i). \end{aligned} \tag{3}$$

[1] https://github.com/HLT-ISTI/QuaPy/tree/lequa2022/LeQua2022

The difference between EDy and the method introduced in [6] is how to compute $\delta(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The authors in [6] propose to use the actual features of the input space. We denote such approach as EDX. Our proposal is to use the predictions of a classifier, h. In symbols, $\delta(h(\boldsymbol{x}_i), h(\boldsymbol{x}_j))$, the same predictions used by EM and HDy. As function δ we used the Manhattan distance.

3. Experiments

The first aspect of our experiments[2] was to select the best classifier because all the compared algorithms require a classifier. We tested several classifiers using just the training set of subtask *T1A*, including Logistic Regression, Random Forest, Support Vector Machines, XGboost, Naive Bayes and Gaussian processes. The best one was Logistic Regression (LR) rather clearly. Then we adjusted its regularization parameter resulting than the best value was $C = 0.01$. We employed this classifier for the rest of the experiments including subtask $T1B$.

Another important factor according to our experience is how to estimate the distributions. It is well-described in the literature, for instance for AC method [7], that it is better to use some sort of cross-validation (CV). Our approach in our recent papers is to use such CV to estimate both, the distributions of the training data, but also for the testing sets, averaging the predictions of the classifiers that compose the CV model. This works better that learning a separated classifier to estimate any value of the testing bags. We used 20 folds for subtask *T1A* and 10 for *T1B*.

The only compared method that has hyperparameters is HDy:

- Similarity Measure. Two alternatives: HD, as the original method HDy, and Topsøe (method DyS-TS in [5]). We will denoted this last method as $PDFyT$, because it uses histograms (PDF), the predictions from a classifier (y) and Topsøe (T).
- Number of bins. We tried the following group of values $\{30, 40, 50\}$.
- Method used for computing the cut points for the histograms: equal-width or equal-count.

We just tried these six choices to select the best combination for HDy and $PDFyT$.

Recall that the target performance measure is the Mean of the Relative Absolute Error (MRAE):

$$MRAE(p, \hat{p}) = \frac{1}{|k|} \sum_{l=1}^{k} \frac{|\hat{p}_l - p_l|}{p_l}, \tag{4}$$

where p_l and $\hat{p}_l$ are the real and the predicted prevalences for class l. RAE may be undefined when $p_l = 0$, so both prevalences are smoothed before computing it [8]:

$$smooth(p) = \frac{\epsilon + p}{\epsilon k + 1}, \quad \epsilon = \frac{1}{2m}. \tag{5}$$

3.1. Subtask *T1A*

For this task the equal-count method works better. The results using LR over the validation set are those in Table 1. The first conclusion is that HDy and $PDFyT$ are the best performers. There is no much difference between them but $PDFyT$ seems slightly better. This is in line with the conclusions in [5].

[2]Source code: https://github.com/jjdelcoz/QU-Ant

Table 1
Results over the validation set of subtask *T1A* using Logistic Regression

Method	MRAE	MAE
EM	1.19731	0.22649
HDy (30 bins)	0.15273	0.02570
HDy (40 bins)	0.13941	0.02639
HDy (50 bins)	0.14917	0.02748
$PDFyT$ (30 bins)	0.13542	**0.02411**
$PDFyT$ (40 bins)	0.13225	0.02480
$PDFyT$ (50 bins)	**0.13112**	0.02469
EDy	0.21878	0.02676

Table 2
Results over the validation set of *T1A* using Calibrated Logistic Regression

Method	MRAE	MAE
EM	0.13775	0.02374
HDy (30 bins)	0.18334	0.03077
HDy (40 bins)	0.19601	0.03561
HDy (50 bins)	0.20383	0.04044
$PDFyT$ (30 bins)	0.13025	**0.02425**
$PDFyT$ (40 bins)	**0.12701**	0.02470
$PDFyT$ (50 bins)	0.12825	0.02552
EDy	0.21586	0.02692

EDy is clearly outperformed in terms of MRAE, but its performance is similar to HDy in terms of MAE. Moreover, it is pretty clear from the results in Table 1 that the scores of EM are rather bad because it requires well-calibrated posterior probabilities. Thus, we used the CalibratedCV object of sklearn to obtain calibrated probabilities. The scores of such experiment are in Table 2. EM clearly improves but it performs worse than $PDFyT$. Notice that the score of EM is just slightly better than that provided by the organizers (0.13775 vs. 0.1393).

Taking into account all these results, we finally selected $PDFyT$ with 40 bins of equal-count using Calibrated Logistic Regression with $C = 0.01$. Notice that $PDFyT$ obtains better results that the original version of HDY provided by the organizers (0.12701 vs. 0.1767).

3.2. Subtask *T1B*

Due to the lack of time, we just tried here basically the same configuration of the classifier selected for subtask *T1A* in combination with the OneVsRestClassifier provided by sklearn. The only changes were: i) we had to reduce the number of folds for the cross-validation used to 10 folds because the smallest class has 14 examples, and ii) for HDy the best bin strategy was equal-width and the number of bins tested were $\{4, 8, 16\}$ because the performance tended to decrease as the number of bins increased in this case. Notice that $PDFyT$ could not be employed here because it uses search (not optimization) for computing the final prevalences

Table 3
Results over the validation set of *T1B* using OVR(Calibrated LR) with $\epsilon = 0.002$ (incorrect)

Method	MRAE	MAE
EM	0.74921	0.01637
HDy (4 bins)	0.86716	0.01527
HDy (8 bins)	0.80127	0.01402
HDy (16 bins)	0.85876	0.01586
EDy	**0.68223**	**0.01173**

Table 4
Results over the validation set of *T1B* using OVR(Calibrated LR) with $\epsilon = 0.0005$ (correct)

Method	MRAE	MAE
EM	**1.12322**	0.01637
HDy (4 bins)	1.47463	0.01527
HDy (8 bins)	1.33846	0.01402
HDy (16 bins)	1.39885	0.01586
EDy	1.16777	**0.01173**

Table 5
Results over the validation set of *T1B* using Logistic Regression with $\epsilon = 0.0005$ (correct)

Method	MRAE	MAE
EM	2.35675	0.02811
HDy (4 bins)	0.95555	0.01158
HDy (8 bins)	1.07063	0.01257
HDy (16 bins)	1.19310	0.01494
EDy	**0.89837**	**0.00996**

(exhaustive search is not suitable because the searching space is $[0, 1]^{28}$ and other methods that should have been implemented, such as genetic algorithms, do not guarantee to find the optimal solution). When we performed this experiment we committed a terrible mistake: the value used for the parameter ϵ of MRAE was 0.002 (the one for substask *T1A*), instead of the correct value 0.0005. The results of such incorrect experiment are in Table 3. In such circumstances, EDy seemed the best method: its performance was much better than the rest of approaches, including the baselines provided by the organizers and the results of HistNet (the method of the other team from the U. of Oviedo). Thus we submitted the predictions of EDy to the competition.

But the problem was of course the value of ϵ. The results over the validation set using the correct value are in Table 4. In this case, EM performs better in terms of MRAE but worse than EDy for MAE. In both cases, their results are worse than those of the two best competitors.

After exchanging some emails with TU Dortmund University team, we did one last experiment. Instead of using OneVsRestClassifier and Calibrated Logistic Regression we just applied a plain Logistic Classifier in combination with the same cross-validation estimation procedure (10 folds). The results of EDY would improve significantly, see Table 5. Also the scores of HDy

are very competitive, while those of EM are much worse as it occurred in subtask *T1A* when the posteriors were not calibrated. If we had sent the predictions of this version of EDy the scores over the test set would have been MRAE 0.864787, MAE 0.00994 which are better than those of the winning team of the competition (MRAE 0.879870, MAE 0.011733).

3.3. Conclusions

We have drawn several interesting conclusions from our participation in LeQua:

1. To obtain good results with quantification algorithms that rely on the use of a classifier it is crucial to select the best classifier-quantifier combination. Obviously, not always the same classifier is the most appropriate one for all quantification algorithms.
2. This implies that quantification competitions are even more complex than classification ones. There are more elements to be adjusted: select a combination of a classifier and a quantifier and adjust their hyperparameters. The search space is sometimes doubled.
3. EM is a very good quantification algorithm but is very sensitive to the classifier calibration. Other methods are more robust in this sense and work well with more classifiers.
4. EDy seems a good approach for multiclass quantification.

Acknowledgments

This research was funded by MINECO (Ministerio de Economía y Competitividad) and FEDER (Fondo Europeo de Desarrollo Regional), grant PID2019-110742RB-I00 (MINECO/FEDER).

References

[1] V. González-Castro, R. Alaiz-Rodríguez, E. Alegre, Class distribution estimation based on the hellinger distance, Information Sciences 218 (2013) 146–164.
[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.
[3] 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.
[4] A. Moreo, A. Esuli, F. Sebastiani, Quapy: A python-based framework for quantification, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 4534–4543.
[5] A. Maletzke, D. dos Reis, E. Cherman, G. Batista, Dys: A framework for mixture models in quantification, in: Proceedings of the AAAI, volume 33, 2019, pp. 4552–4560.
[6] H. Kawakubo, M. C. Du Plessis, M. Sugiyama, Computationally efficient class-prior estimation under class balance change using energy distance, IEICE Tran. on Inf. and Sys. 99 (2016) 176–186.
[7] G. Forman, Quantifying counts and costs via classification, Data Mining and Knowledge Discovery 17 (2008) 164–206.
[8] F. Sebastiani, Evaluation measures for quantification: an axiomatic approach, Information Retrieval Journal 23 (2020) 255–288.