

Active Learning and the Saerens-Latinne-Decaestecker Algorithm: An Evaluation

Alessio Molinari^{1,*}, Andrea Esuli¹ and Fabrizio Sebastiani¹

¹*Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, 56124, Pisa, Italy*

Abstract

The Saerens-Latinne-Decaestecker (SLD) algorithm is a method whose goal is improving the quality of the posterior probabilities (or simply “posteriors”) returned by a probabilistic classifier in scenarios characterized by *prior probability shift* (PPS) between the training set and the unlabelled (“test”) set. This is an important task, (a) because posteriors are of the utmost importance in downstream tasks such as, e.g., multiclass classification and cost-sensitive classification, and (b) because PPS is ubiquitous in many applications. In this paper we explore whether using SLD can indeed improve the quality of posteriors returned by a classifier trained via *active learning* (AL), a class of machine learning (ML) techniques that indeed tend to generate substantial PPS. Specifically, we target *AL via relevance sampling* (ALvRS) and *AL via uncertainty sampling* (ALvUS), two AL techniques that are very well-known especially because, due to their low computational cost, are suitable to being applied in scenarios characterized by large datasets. We present experimental results obtained on the RCV1-v2 dataset, showing that SLD fails to deliver better-quality posteriors with both ALvRS and ALvUS, thus contradicting previous findings in the literature, and that this is due not to the amount of PPS that these techniques generate, but to how the examples they prioritize for annotation are distributed.

Keywords

Text Classification, Probabilistic Classifiers, Active Learning, Posterior Probabilities, Prior Probabilities, Prior Probability Shift, Dataset Shift

1. Introduction

In the field of probabilistic classification, a *posterior probability* (or simply: a *posterior*) $\Pr(y|\mathbf{x})$ represents the confidence that a classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ has in the fact that an unlabelled (“test”) document \mathbf{x} belongs to class y . As all confidence scores, posteriors are useful for ranking unlabelled documents (say, in terms of perceived relevance to class y). However, for some downstream tasks other than ranking, such as multiclass classification and cost-sensitive classification, standard (non-probabilistic) confidence scores are not enough, and true posteriors are needed.


For these downstream tasks to be carried out accurately, it is essential that the posteriors are


CIRCLE 2022: 2nd Joint Conference of the Information Retrieval Communities in Europe, July 4-7, 2022, Samatan, FR

*Corresponding author.

✉ alessio.molinari@isti.cnr.it (A. Molinari); andrea.esuli@isti.cnr.it (A. Esuli); fabrizio.sebastiani@isti.cnr.it (F. Sebastiani)

ORCID 0000-0002-8791-3245 (A. Molinari); 0000-0002-5725-4322 (A. Esuli); 0000-0003-4221-6427 (F. Sebastiani)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

high-quality, i.e., well-calibrated.¹ Some classifiers (e.g., those trained by logistic regression) tend to return calibrated posteriors (we thus say that they are *calibrated classifiers*); some other classifiers (e.g., those trained by naive Bayesian methods) tend to return posteriors that are not calibrated; yet some other classifiers return confidence scores that are not probabilities. For the last two cases, methods exist (see e.g., [1, 2]) to calibrate uncalibrated classifiers.

Unfortunately, independently of the learning method used for training the classifiers, posteriors tend to be uncalibrated when the application scenario suffers from *prior probability shift* (PPS – [3]), i.e., the (ubiquitous) phenomenon according to which the distribution $p_U(y)$ of the unlabelled test documents U across the classes is different from the distribution $p_L(y)$ of the labelled training documents L . This is due to the fact that when the (calibrated or uncalibrated) classifiers generate the posteriors, they assume that the class prior probabilities $p_U(y)$ (a.k.a. “priors”, or “class prevalence values”) in the set U of unlabelled documents are the same as those encountered in the training set L . If this is not the case, the returned posteriors end up not being calibrated.

The Saerens-Latinne-Decaestecker (SLD) algorithm [4] is a well-known method for recalibrating the posteriors of a set of unlabelled documents in the presence of PPS between the training set and this latter set. Given a machine-learned classifier and a set of unlabelled documents for which the classifier has returned posteriors and estimates of the priors, SLD updates them both in an iterative, mutually recursive way, with the goal of making both more accurate. Since its publication, SLD has become the standard algorithm for recalibrating the posteriors in the presence of PPS, and is still considered a top contender (see [5]) when we need to estimate the priors (a task that has become known as “quantification”).

However, its real effectiveness in improving the quality of the posteriors is not yet entirely clear. On one side, a recent, large experimental study [6] has shown that, at least when the number of classes in the classification scheme is very small and the classifier is calibrated, SLD does improve the quality of the posteriors, and especially so when the amount of PPS is high. On another side, in experiments aimed at improving the quality of cost-sensitive text classification in *technology-assisted review* (TAR) [7, 8, 9], SLD has (strangely) not delivered any measurable improvement in the quality of the posteriors, not even when the amount of PPS was high [10, 11]. The relationship between SLD and PPS is thus still unclear.

The goal of this paper is to shed some light on this relationship. The reason why we are interested in this is that, if SLD indeed improved the quality of the posteriors under PPS, it would be extremely useful for TAR. In fact, in TAR we typically use a classifier trained on labelled data in order to return posterior probabilities of relevance for a large set of unlabelled documents. These posteriors are needed for ranking the unlabelled documents in terms of their probability of relevance, and high-quality posteriors are of key importance for approaches to

¹The posteriors $\Pr(y|\mathbf{x})$, where \mathbf{x} belongs to a set $\sigma = \{\mathbf{x}_1, \dots, \mathbf{x}_{|\sigma|}\}$, are said to be *well-calibrated* when, for all $a \in [0, 1]$, it holds that

$$\frac{|\{\mathbf{x} \in \sigma \cap y \mid \Pr(y|\mathbf{x}) = a\}|}{|\{\mathbf{x} \in \sigma \mid \Pr(y|\mathbf{x}) = a\}|} \approx a \quad (1)$$

Perfect calibration is usually unattainable on any non-trivial dataset; however, calibration comes in degrees (and the quality of calibration can indeed be measured), so efforts can be made to obtain posteriors which are as close as possible to their perfectly calibrated counterparts.

TAR based on risk minimization [9]. Additionally, TAR settings are typically characterized by PPS, because the typical way to build a training set L in TAR is via *active learning* (AL), which usually generates PPS. So, the research question we want to answer is

RQ: Does SLD improve the quality of posterior probabilities in situations in which the training set L used for training the probabilistic classifier has been generated via active learning?

In the rest of the paper we briefly introduce the SLD algorithm (Section 2) and the two active learning techniques (ALvRS and ALvUS) we use in order to investigate our research question (Section 3), after which we present the results of our experiments (Section 4) followed (Section 5) by a few concluding remarks.

2. The SLD Algorithm

We assume a training set L of labelled examples and a set $U = \{(\mathbf{x}_1, t(\mathbf{x}_1)), \dots, (\mathbf{x}_{|U|}, t(\mathbf{x}_{|U|}))\}$ of unlabelled examples, i.e., examples whose true labels $t(\mathbf{x}) \in \mathcal{Y} = \{y_1, \dots, y_{|\mathcal{Y}|}\}$ are unknown to the system.

SLD, proposed by Saerens et al. [4], is an instance of Expectation Maximization [12], a well-known iterative algorithm for finding maximum-likelihood estimates of parameters (in our case: the class prior probabilities) for models that depend on unobserved variables (in our case: the class labels). Pseudocode of the SLD algorithm is here included as Algorithm 1.

Essentially, SLD iteratively updates (Line 13) the estimates of the class priors by using the posteriors computed in the previous iteration, and updates (Line 15) the posteriors by using the estimates of the class priors computed in the present iteration, in a mutually recursive fashion. The main goal is to adjust the posteriors and re-estimate the priors in such a way that they are mutually consistent, i.e., that they should be such that

$$\Pr_U(y) = \frac{1}{|U|} \sum_{\mathbf{x} \in U} \Pr(y|\mathbf{x}) \quad (2)$$

Equation 2 is a necessary (albeit not sufficient) condition for the posteriors $\Pr(y|\mathbf{x})$ of the documents $\mathbf{x} \in U$ to be calibrated. SLD may thus be viewed as making a step towards calibrating these posteriors.

The algorithm iterates until convergence, i.e., until the class priors become stable and Equation 2 is satisfied. The convergence of SLD may be tested by computing how the distribution of the priors at iteration $(s - 1)$ and that at iteration (s) still diverge; this can be evaluated, for instance, in terms of absolute error, i.e.,²

$$\text{AE}(\hat{p}_U^{(s-1)}, \hat{p}_U^{(s)}) = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} |\hat{\Pr}_U^{(s)}(y) - \hat{\Pr}_U^{(s-1)}(y)| \quad (3)$$

²Consistently with most mathematical literature, we use the caret symbol ($\hat{\cdot}$) to indicate estimation.

Algorithm 1: The SLD algorithm [4].

Input : Class priors $\Pr_L(y)$ on L , for all $y \in \mathcal{Y}$;
Posterior probabilities $\Pr(y|\mathbf{x})$, for all $y \in \mathcal{Y}$ and for all $\mathbf{x} \in U$;
Output: Estimates $\hat{\Pr}_U(y)$ of class prevalence values on U , for all $y \in \mathcal{Y}$;
Updated posterior probabilities $\Pr(y|\mathbf{x})$, for all $y \in \mathcal{Y}$ and for all $\mathbf{x} \in U$;

```
1 // Initialization
2  $s \leftarrow 0$ ;
3 for  $y \in \mathcal{Y}$  do
4    $\hat{\Pr}_U^{(s)}(y) \leftarrow \Pr_L(y)$ ; // Initialize the prior estimates
5   for  $\mathbf{x} \in U$  do
6      $\Pr^{(s)}(y|\mathbf{x}) \leftarrow \Pr(y|\mathbf{x})$ ; // Initialize the posteriors
7   end
8 end
9 // Main Iteration Cycle
10 while stopping condition = false do
11    $s \leftarrow s + 1$ ;
12   for  $y \in \mathcal{Y}$  do
13      $\hat{\Pr}_U^{(s)}(y) \leftarrow \frac{1}{|U|} \sum_{\mathbf{x} \in U} \Pr^{(s-1)}(y|\mathbf{x})$ ; // Update the prior estimates
14     for  $\mathbf{x} \in U$  do
15        $\Pr^{(s)}(y|\mathbf{x}) \leftarrow \frac{\hat{\Pr}_U^{(s)}(y) \cdot \Pr^{(0)}(y|\mathbf{x})}{\sum_{y \in \mathcal{Y}} \frac{\hat{\Pr}_U^{(s)}(y)}{\hat{\Pr}_U^{(0)}(y)} \cdot \Pr^{(0)}(y|\mathbf{x})}$  // Update the posteriors
16     end
17   end
18 end
19 // Generate output
20 for  $y \in \mathcal{Y}$  do
21    $\hat{\Pr}_U(y) \leftarrow \hat{\Pr}_U^{(s)}(y)$ ; // Return the prior estimates
22   for  $\mathbf{x} \in U$  do
23      $\Pr(y|\mathbf{x}) \leftarrow \Pr^{(s)}(y|\mathbf{x})$  // Return the adjusted posteriors
24   end
25 end
```

In the experiments of Section 4, we decree that convergence has been reached when $\text{AE}(\hat{p}_U^{(s-1)}, \hat{p}_U^{(s)}) < 10^{-6}$; we stop SLD when we have reached either convergence or the maximum number of iterations (that we set to 1000).

While SLD is a natively multiclass algorithm, in this paper we restrict our analysis to the binary case, with codeframe $\mathcal{Y} = \{\oplus, \ominus\}$.

3. Active learning policies

In the experiments for this work, we test the SLD algorithm on training/test sets generated via two of the best-known active learning policies, namely *Active Learning via Relevance Sampling* (ALvRS) and *Active Learning via Uncertainty Sampling* (ALvUS), first presented in [13]. While fairly old and unsophisticated, these policies are still very popular because their computational cost is small, which makes them extremely suitable to applications (such as TAR) in which the set of unlabelled documents that are candidates for annotation, and that the AL policy must thus rank, is large.

Active Learning via Relevance Sampling (ALvRS). ALvRS is an interactive process which, given a data pool of unlabelled documents P , asks the reviewer to annotate an initial “seed” set of documents $S \subset P$, uses S as the training set L to train a binary classifier h , and uses h to rank the documents in $(P \setminus L)$ in decreasing order of their posterior probability of relevance $\Pr(\oplus|\mathbf{x})$. Then, the reviewer is asked to annotate the b documents for which $\Pr(\oplus|\mathbf{x})$ is highest (with b the *batch size*), which, once annotated, are added to the training set L . Finally, we retrain our classifier on the new training set and repeat the process, until a predefined number of documents (the *annotation budget*) have been reviewed.

Active Learning via Uncertainty Sampling (ALvUS). The ALvUS policy is a variation of ALvRS, where we review the documents not in decreasing order of $\Pr(\oplus|\mathbf{x})$ but in decreasing order of $|\Pr(\oplus|\mathbf{x}) - 0.5|$, i.e., we top-rank the documents which the classifier is most uncertain about.

The *Rand* policies. For each of the two policies defined above, we define an “oracle-like” policy which we will use for a control experiment. The aim of these policies, which we call *Rand*(RS) and *Rand*(US) (corresponding to ALvRS and ALvUS, respectively), is helping to better understand whether the results we are seeing are due to the PPS generated by the two active learning policies, or to their document selection strategy. Given a set L of labelled documents and a set U of unlabelled documents generated via an active learning policy by sampling P , the *Rand* policy samples P randomly to generate alternative labelled and unlabelled sets L' and U' subject to the constraints that $|L| = |L'|$, $|U| = |U'|$, $\Pr_L(\oplus) = \Pr_{L'}(\oplus)$, and $\Pr_U(\oplus) = \Pr_{U'}(\oplus)$. In other words, the *Rand* policies generates the same PPS as the active learning policy, but with a different choice of documents.

4. Experiments

We run a set of comparative experiments to explore the interaction between AL-based classifiers and SLD. In order to do this we test the two AL policies described above, ALvRS and ALvUS, and compare them with the *Rand*(RS) and *Rand*(US) policies.

4.1. The RCV1-v2 dataset

We run our experiments on the RCV1-v2 dataset [14], a multi-label multi-class collection of 804,414 Reuters news (produced from August 1996 to August 1997)³. The RCV1-v2 codeframe

³We use the RCV1-v2 dataset as provided by the scikit-learn implementation. https://scikit-learn.org/stable/datasets/real_world.html#rcv1-dataset

consists of a set of 103 classes. Since in this work we experiment with binary classification problems only, for each such class y we consider a binary codeframe $\mathcal{Y}^y = \{\oplus, \ominus\}$, where $\oplus = y$ and $\ominus = \bar{y}$. Finally, in order to keep computational costs within reasonable bounds, we only work with a pool P consisting of the first 100,000 documents of the RCV1-v2 collection.

4.2. Experimental setup

For each class $y \in \mathcal{Y}$, and for each AL policy, we run the AL process to generate a sequence of binary classification training sets with incremental sizes; this determines a corresponding sequence of test sets, since the pool P is always the union of the training set L and the test set U . As for training the classifier, in all of our experiments we use a SVM algorithm, post-calibrated via Platt calibration [1].

The active learning process is seeded with a set of 1000 initial training documents $S \subset P$, i.e., we randomly sample 1000 documents from our pool P and train our classifier on it. Since in order to calibrate the SVM classifier we need at least 2 positive instances (i.e., instances of \oplus) for cross-validation, we always ensure this condition is respected in S . We then run the active learning process on the remaining 99,000 documents. This procedure is illustrated in Algorithm 2. As previously mentioned, we also generate an analogous sequence of training/test

Algorithm 2: Pseudo-code to generate active learning datasets.

Input: Documents P ; Set of training set sizes Σ ; AL policy a ; Batch size b

```

1  $L \leftarrow \text{random\_sample}(P, 1000)$ ;
2  $P \leftarrow P - L$ ;
3  $m \leftarrow \max(\Sigma)$ ;
4  $i \leftarrow |L|$ ;
5  $\phi \leftarrow \text{train\_svm}(L)$ ;
6 while  $|L| < m$  do
7    $L \leftarrow L \cup \text{select\_via\_policy}(\phi, a, P, b)$ ;
8    $\phi \leftarrow \text{train\_svm}(L)$ ;
9    $P \leftarrow P - L$ ;
10   $i \leftarrow |L|$ ;
11  if  $i \in \Sigma$  then
12     $\text{save}(L, P)$ 
13  end
14 end

```

sets with a *Rand* policy, i.e., random sampling constrained to keep the same class prevalence values obtained by the corresponding active learning policies.

Once the different training sets are generated, we train a calibrated SVM from scratch on each of them and obtain a set of posterior probabilities $\text{Pr}^{\text{PreSLD}}(\oplus|\mathbf{x})$ for each respective test set. Finally, we apply the SLD algorithm, obtaining a new set of posteriors $\text{Pr}^{\text{PostSLD}}(\oplus|\mathbf{x})$.

In TAR scenarios, we are usually interested on the classification performance on the entire pool P : for this reason, we merge the labels on the training set with the posterior probabilities

on the test set, obtaining a new set of probabilities $\Pr(\oplus|\mathbf{x})$ where, for all $\mathbf{x} \in L$, we take $\Pr(\oplus|\mathbf{x}) = 1$ iff \oplus is the true label of \mathbf{x} and $\Pr(\oplus|\mathbf{x}) = 0$ iff \ominus is the true label of \mathbf{x} , with L the training set. All of our evaluation measures are computed on this set of probabilities.

4.3. Evaluation measures

To evaluate the performance of our classifier and the quality of the posteriors we use several metrics, namely, Accuracy, Precision, Recall, F_1 , and Brier Score. We will explain more in detail the last metric, as the reader is likely familiar with the first four.

Given a set $U = \{(\mathbf{x}_1, t(\mathbf{x}_1)), \dots, (\mathbf{x}_{|U|}, t(\mathbf{x}_{|U|}))\}$ of unlabelled documents to be labelled according to codeframe $\mathcal{Y} = \{\oplus, \ominus\}$, and given posteriors $\Pr(\oplus|\mathbf{x})$ for these documents, the Brier score [15] is defined as

$$\text{BS} = \frac{1}{|U|} \sum_{i=1}^{|U|} (I(t(\mathbf{x}_i) = \oplus) - \Pr(\oplus|\mathbf{x}_i))^2 \quad (4)$$

where $I(\cdot)$ is a function that returns 1 if its argument is true and 0 otherwise. BS ranges between 0 (best) and 1 (worst), i.e., it is a measure of error, and not of accuracy, and rewards probabilistic classifiers that return a high value of $\Pr(\oplus|\mathbf{x})$ for instances of \oplus and a low such value for instances of \ominus . In our result tables we will report, instead of the Brier score, its complement to 1, i.e., $(1 - \text{BS})$, so that all our metrics can be interpreted as “the higher, the better”.

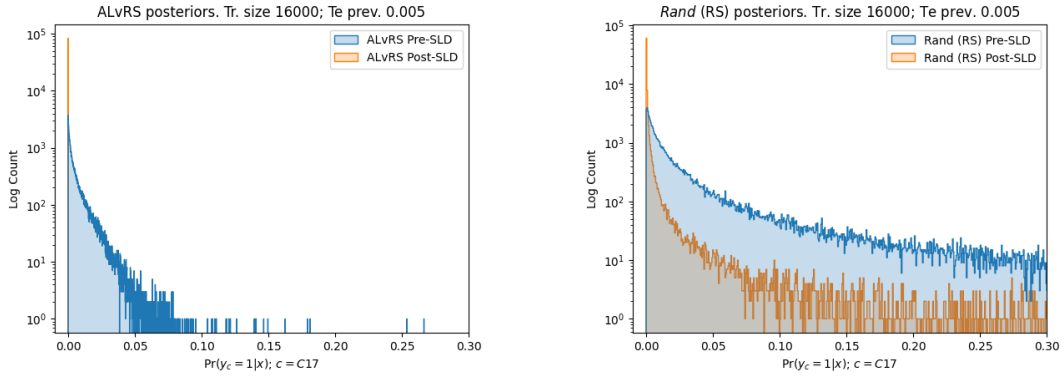
4.4. Results

We present the results of our experiments in Table 1 for ALvRS and in Table 2 for ALvUS.

These results are averages across all of the 103 RCV1-v2 classes used in our experiments. We show both the average results for each training set size (2000, 4000, 8000, 16000) and the results averaged on all sizes. We note that *in all cases (i.e., for both ALvRS and ALvUS, and for all sizes), the use of SLD has a detrimental effect on the posterior probabilities. However, while this is true for the setups generated via active learning, the use of SLD has a beneficial effect on the posteriors when the Rand policies have been used.*

What we see on the AL datasets seems to contradict what was argued in [6], i.e., that SLD can improve posteriors in binary classification contexts with high PPS. The *Rand* policy, which resembles the test data generation technique used in [6, Section 3.2.1], seems instead to confirm the conclusions of [6]. However, when the training and the test sets do not originate from random sampling, as it is the case for the AL datasets, this hypothesis is disconfirmed.

While we defer a proper analysis of the causes of this problem to future work, a first hypothesis might be that the following is happening. When building active learning datasets, we can assume that the documents that remain in the test set, as this decreases in size, are documents for which the classifier is either fairly sure of their negative label (ALvRS) or of their label in general (ALvUS). Furthermore, AL policies such as relevance sampling or uncertainty sampling suffer from *sampling bias* [16], since both AL strategies solely depend on what the classifier thinks is either relevant or uncertain; this means that, as the active learning phase proceeds, the annotator is asked to review documents that are very similar among each other and, because of



(a) ALvRS posteriors pre- and post-SLD.

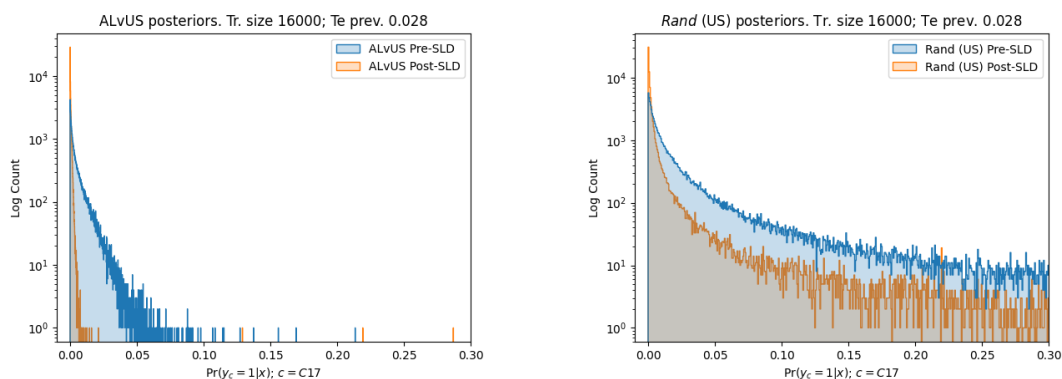
(b) *Rand* posteriors pre- and post-SLD.

Figure 1: Posterior probabilities $\Pr(y = 1|x)$ pre- and post-SLD for ALvRS and *Rand*(RS) (with ALvRS test prevalence values) datasets (training set size = 16,000; class C17). On the y axis we plot the log count of the posteriors for better readability.

this, not enough informative or representative of the actual dataset. Hence, especially when the prevalence of \oplus is very low, we may expect the distribution of the posterior probabilities to be strongly skewed towards the negative class, much more than if the dataset were a random sample of the population (as it is for the two *Rand* policies); in the latter case, the classifier might still find documents in the test set for which its confidence is lower than in the AL case. This can be seen in Figures 1 and 2, where we plot the posteriors $\Pr(\oplus|\mathbf{x})$ pre- and post-SLD for ALvRS, ALvUS and *Rand* on a random RCV1-v2 class used in our experiments (C17, training size 16,000). Notice how in both cases (RS and US), the posteriors distribution on the AL dataset is strongly skewed towards 0, whereas *Rand*'s is slightly more spread on the $[0.0, 0.3]$ interval⁴. SLD seems to perform a correct rescaling of the posteriors in the *Rand* cases, whereas it simply sets all posteriors to 0 in the AL cases. Since the PPS is equivalent in both cases, the reasons are to be found in the document strategy selection, within the SLD algorithm, or both. As we mentioned before, the sampling bias is likely responsible for the skewness of the posterior probability distributions that we see in the plots, as this is the only and major difference between the AL and *Rand* policies. On the other end, if the estimated prevalence $\Pr^{(s)}$ (which we compute as the average of the posteriors, see Algorithm 1) is close to 0, as we see in the figures, then indeed SLD will drag the distribution towards 0. As a matter of fact, consider the SLD update of the prior and posterior probabilities performed in Line 13 and Line 15, respectively, of Algorithm 1. It is trivial to see that $\lim_{\hat{\Pr}_U^{(s)} \rightarrow 0} \Pr^{(s)}(y|\mathbf{x}) = 0$, i.e., the “maximization” of the “expectation” is that there are no positive instances in the AL test set.

All this would require a deeper analysis, which however we defer to future work.

⁴We did not plot the entire $[0.0, 1.0]$ interval as there was hardly any probability after the 0.3 threshold. This makes the plots more readable.



(a) ALvUS posteriors pre- and post-SLD.

(b) *Rand* posteriors pre- and post-SLD.

Figure 2: Posterior probabilities $\Pr(y = 1|x)$ pre- and post-SLD for ALvUS and *Rand*(US) (with ALvUS test prevalence values) datasets (training set size = 16,000; class C17). On the y axis we plot the log count of the posteriors for better readability.

5. Conclusion

We have studied the interactions between active learning methods and the SLD algorithm. It is known that AL-generated scenarios tend to exhibit a high prior probability shift, and that in past research SLD has proven effective in improving the quality of the posteriors on sets of unlabelled data, especially in cases of high PPS. We thus tested the use of SLD on the posteriors generated by classifiers trained on AL-generated training sets, testing the hypothesis that SLD would improve the quality of these posteriors. Our results do not support this hypothesis, showing instead that the posteriors returned by AL-based classifiers deteriorate after the application of SLD. We have run control experiments that used the same amount of PPS of the AL-generated scenarios, albeit obtained by sampling the elements of the pool randomly. In this case SLD did improve the quality of the posteriors, which indicates that SLD has a specific problem not with the amount of PPS but with the documents selected by AL techniques.

From these preliminary experiments we conclude that, counterintuitively, it is not recommended to combine AL and SLD. In future work we will investigate more deeply the causes of this problem, i.e., what aspect of the AL process results in the bad interaction with SLD, and if and how it is possible to solve this problem, so as to combine the benefits of both methods.

Acknowledgments

This work has been supported by the AI4Media project, funded by the European Commission (Grant 951911) under the H2020 Programme ICT-48-2020, and by the SoBigData++ project, funded by the European Commission (Grant 871042) under the H2020 Programme INFRAIA-2019-1. The authors' opinions do not necessarily reflect those of the European Commission.

References

- [1] 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.
- [2] B. Zadrozny, C. Elkan, Transforming classifier scores into accurate multiclass probability estimates, in: *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (KDD 2002)*, Edmonton, CA, 2002, pp. 694–699. doi:10.1145/775107.775151.
- [3] 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.
- [4] 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.
- [5] A. Moreo, F. Sebastiani, Tweet sentiment quantification: An experimental re-evaluation, *PLoS ONE* (2022). Forthcoming.
- [6] 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.
- [7] M. R. Grossman, G. V. Cormack, Technology-assisted review in e-discovery can be more effective and more efficient than exhaustive manual review, *Richmond Journal of Law and Technology* 17 (2011) Article 5.
- [8] D. W. Oard, J. R. Baron, B. Hedin, D. D. Lewis, S. Tomlinson, Evaluation of information retrieval for E-discovery, *Artificial Intelligence and Law* 18 (2010) 347–386.
- [9] D. W. Oard, F. Sebastiani, J. K. Vinjumur, Jointly minimizing the expected costs of review for responsiveness and privilege in e-discovery, *ACM Transactions on Information Systems* 37 (2018) 11:1–11:35. doi:10.1145/3268928.
- [10] A. Molinari, Leveraging the transductive nature of e-discovery in cost-sensitive technology-assisted review, in: *Proceedings of the 8th BCS-IRSG Symposium on Future Directions in Information Access (FDIA 2019)*, Milano, IT, 2019, pp. 72–78.
- [11] A. Molinari, Risk minimization models for technology-assisted review and their application to e-discovery, Master’s thesis, Department of Computer Science, University of Pisa, Pisa, IT, 2019.
- [12] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society, B* 39 (1977) 1–38.
- [13] D. D. Lewis, W. A. Gale, A sequential algorithm for training text classifiers, in: *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval (SIGIR 1994)*, Dublin, IE, 1994, pp. 3–12. doi:10.1007/978-1-4471-2099-5_1.
- [14] D. D. Lewis, Y. Yang, T. G. Rose, F. Li, RCV1: A new benchmark collection for text categorization research, *Journal of Machine Learning Research* 5 (2004) 361–397.
- [15] G. W. Brier, Verification of forecasts expressed in terms of probability, *Monthly Weather Review* 78 (1950) 1–3. doi:10.1175/1520-0493(1950)078<0001:vofeit>2.0.co;2.
- [16] S. Dasgupta, D. Hsu, Hierarchical sampling for active learning, in: *Proceedings of the 25th International Conference on Machine Learning (ICML 2018)*, Stockholm, SE, 2008, pp. 208–215.

	ALvRS		<i>Rand</i> (RS)	
	Pre-SLD	Post-SLD	Pre-SLD	Post-SLD
Training set size: 2000. $\Pr_L(\oplus)$: 0.275 ± 0.196 ; $\Pr_{(P \setminus L)}(\oplus)$: 0.027 ± 0.062				
Accuracy	0.988 ± 0.018	0.981 ± 0.030	0.974 ± 0.026	0.989 ± 0.015
Precision	0.930 ± 0.053	0.999 ± 0.005	0.405 ± 0.166	0.734 ± 0.171
Recall	0.630 ± 0.194	0.472 ± 0.243	0.926 ± 0.102	0.800 ± 0.142
F1	0.735 ± 0.160	0.604 ± 0.228	0.548 ± 0.157	0.755 ± 0.151
(1-BS)	0.990 ± 0.014	0.982 ± 0.025	0.980 ± 0.020	0.992 ± 0.011
Training set size: 4000. $\Pr_L(\oplus)$: 0.315 ± 0.020 ; $\Pr_{(P \setminus L)}(\oplus)$: 0.020 ± 0.059				
Accuracy	0.991 ± 0.017	0.984 ± 0.043	0.969 ± 0.038	0.992 ± 0.014
Precision	0.976 ± 0.040	1.000 ± 0.001	0.434 ± 0.134	0.809 ± 0.119
Recall	0.817 ± 0.145	0.754 ± 0.217	0.974 ± 0.057	0.906 ± 0.098
F1	0.884 ± 0.104	0.840 ± 0.164	0.590 ± 0.122	0.850 ± 0.096
(1-BS)	0.992 ± 0.013	0.985 ± 0.035	0.976 ± 0.029	0.994 ± 0.001
Training set size: 8000. $\Pr_L(\oplus)$: 0.242 ± 0.266 ; $\Pr_{(P \setminus L)}(\oplus)$: 0.013 ± 0.053				
Accuracy	0.994 ± 0.015	0.988 ± 0.048	0.974 ± 0.042	0.995 ± 0.012
Precision	0.993 ± 0.022	1.000 ± 0.000	0.538 ± 0.121	0.874 ± 0.096
Recall	0.908 ± 0.091	0.882 ± 0.158	0.983 ± 0.048	0.947 ± 0.070
F1	0.947 ± 0.059	0.927 ± 0.120	0.688 ± 0.100	0.905 ± 0.070
(1-BS)	0.995 ± 0.012	0.988 ± 0.046	0.980 ± 0.031	0.996 ± 0.009
Training set size: 16000. $\Pr_L(\oplus)$: 0.154 ± 0.215 ; $\Pr_{(P \setminus L)}(\oplus)$: 0.008 ± 0.042				
Accuracy	0.997 ± 0.013	0.993 ± 0.035	0.983 ± 0.036	0.997 ± 0.010
Precision	0.997 ± 0.012	1.000 ± 0.000	0.666 ± 0.105	0.908 ± 0.097
Recall	0.953 ± 0.055	0.942 ± 0.099	0.987 ± 0.038	0.970 ± 0.046
F1	0.974 ± 0.033	0.967 ± 0.067	0.791 ± 0.080	0.935 ± 0.069
(1-BS)	0.997 ± 0.010	0.993 ± 0.035	0.987 ± 0.027	0.997 ± 0.007
Average across all training set sizes. $\Pr_L(\oplus)$: 0.247 ± 0.068 ; $\Pr_{(P \setminus L)}(\oplus)$: 0.017 ± 0.008				
Accuracy	0.992 ± 0.004	0.986 ± 0.005	0.975 ± 0.006	0.993 ± 0.003
Precision	0.974 ± 0.031	1.000 ± 0.000	0.511 ± 0.118	0.831 ± 0.077
Recall	0.827 ± 0.143	0.762 ± 0.209	0.968 ± 0.028	0.906 ± 0.075
F1	0.885 ± 0.107	0.835 ± 0.162	0.654 ± 0.109	0.861 ± 0.079
(1-BS)	0.994 ± 0.003	0.987 ± 0.005	0.981 ± 0.005	0.995 ± 0.003

Table 1

Results for the datasets generated via the ALvRS policy and the *Rand*(RS) policy. Every block in the table refers to a different size of the training set. The last block reports the average values of the blocks above.

	ALvUS		<i>Rand</i> (US)	
	Pre-SLD	Post-SLD	Pre-SLD	Post-SLD
Training set size: 2000. $\Pr_L(\oplus)$: 0.177 ± 0.115 ; $\Pr_{(P \setminus L)}(\oplus)$: 0.029 ± 0.063 .				
Accuracy	0.990 ± 0.014	0.985 ± 0.019	0.980 ± 0.019	0.989 ± 0.015
Precision	0.897 ± 0.071	0.990 ± 0.044	0.459 ± 0.198	0.716 ± 0.172
Recall	0.673 ± 0.187	0.433 ± 0.198	0.893 ± 0.097	0.767 ± 0.131
F1	0.753 ± 0.148	0.575 ± 0.190	0.586 ± 0.177	0.730 ± 0.148
(1-BS)	0.991 ± 0.012	0.987 ± 0.015	0.985 ± 0.015	0.991 ± 0.011
Training set size: 4000. $\Pr_L(\oplus)$: 0.200 ± 0.140 ; $\Pr_{(P \setminus L)}(\oplus)$: 0.025 ± 0.062 .				
Accuracy	0.994 ± 0.010	0.990 ± 0.016	0.981 ± 0.021	0.991 ± 0.013
Precision	0.978 ± 0.031	0.996 ± 0.008	0.493 ± 0.171	0.798 ± 0.114
Recall	0.857 ± 0.127	0.741 ± 0.206	0.955 ± 0.060	0.876 ± 0.105
F1	0.909 ± 0.087	0.831 ± 0.157	0.634 ± 0.143	0.829 ± 0.093
(1-BS)	0.994 ± 0.009	0.991 ± 0.014	0.985 ± 0.016	0.993 ± 0.010
Training set size: 8000. $\Pr_L(\oplus)$: 0.154 ± 0.138 ; $\Pr_{(P \setminus L)}(\oplus)$: 0.021 ± 0.061 .				
Accuracy	0.997 ± 0.006	0.995 ± 0.011	0.985 ± 0.018	0.993 ± 0.011
Precision	0.993 ± 0.010	0.997 ± 0.006	0.589 ± 0.149	0.856 ± 0.090
Recall	0.927 ± 0.080	0.882 ± 0.129	0.967 ± 0.050	0.915 ± 0.075
F1	0.957 ± 0.049	0.930 ± 0.082	0.721 ± 0.114	0.880 ± 0.064
(1-BS)	0.997 ± 0.006	0.995 ± 0.009	0.988 ± 0.014	0.995 ± 0.008
Training set size: 16000. $\Pr_L(\oplus)$: 0.101 ± 0.109 ; $\Pr_{(P \setminus L)}(\oplus)$: 0.018 ± 0.059				
Accuracy	0.999 ± 0.003	0.998 ± 0.003	0.990 ± 0.012	0.995 ± 0.009
Precision	0.996 ± 0.005	0.997 ± 0.005	0.697 ± 0.122	0.888 ± 0.090
Recall	0.960 ± 0.049	0.953 ± 0.060	0.972 ± 0.040	0.941 ± 0.054
F1	0.977 ± 0.029	0.973 ± 0.035	0.806 ± 0.086	0.910 ± 0.064
(1-BS)	0.999 ± 0.003	0.998 ± 0.003	0.992 ± 0.010	0.996 ± 0.007
Avg. across all training set sizes. $\Pr_L(\oplus)$: 0.158 ± 0.043 ; $\Pr_{(P \setminus L)}(\oplus)$: 0.023 ± 0.004				
Accuracy	0.995 ± 0.004	0.992 ± 0.006	0.984 ± 0.005	0.992 ± 0.003
Precision	0.966 ± 0.047	0.995 ± 0.004	0.560 ± 0.107	0.814 ± 0.076
Recall	0.854 ± 0.128	0.752 ± 0.230	0.947 ± 0.037	0.875 ± 0.077
F1	0.899 ± 0.101	0.827 ± 0.179	0.686 ± 0.097	0.837 ± 0.079
(1-BS)	0.995 ± 0.003	0.993 ± 0.005	0.987 ± 0.004	0.994 ± 0.002

Table 2

Results for the datasets generated via the ALvUS policy and the *Rand*(US) policy. Every block in the table refers to a different size of the training set. The last block reports the average values of the blocks above.