Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

Improved risk minimization algorithms for technology-assisted review

Alessio Molinari [a,b,*], Andrea Esuli [a], Fabrizio Sebastiani [a]

[a] *Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Via Giuseppe Moruzzi 1, Pisa, 56124, Italy*
[b] *Dipartimento di Informatica, Università di Pisa, Largo Bruno Pontecorvo 3, 56127, Pisa, Italy*

ARTICLE INFO

Keywords:
Technology-assisted review
E-discovery
Risk minimization
Dataset shift
MINECORE

ABSTRACT

MINECORE is a recently proposed decision-theoretic algorithm for technology-assisted review that attempts to minimise the expected costs of review for responsiveness and privilege in e-discovery. In MINECORE, two probabilistic classifiers that classify documents by responsiveness and by privilege, respectively, generate posterior probabilities. These latter are fed to an algorithm that returns as output, after applying risk minimization, two ranked lists, which indicate exactly which documents the annotators should review for responsiveness and which documents they should review for privilege. In this paper we attempt to find out if the performance of MINECORE can be improved (a) by using, for the purpose of training the two classifiers, active learning (implemented either via relevance sampling, or via uncertainty sampling, or via a combination of them) instead of passive learning, and (b) by using the Saerens-Latinne-Decaestecker algorithm to improve the quality of the posterior probabilities that MINECORE receives as input. We address these two research questions by carrying out extensive experiments on the RCV1-v2 benchmark. We make publicly available the code and data for reproducing all our experiments.

1. Introduction

In several subfields of data science the term "review" refers to the activity, carried out by human annotators (also called *reviewers*, or *coders*), of assigning a textual document $\mathbf{x}$ to a class y, where this class belongs to a finite, predefined set $\mathcal{Y} = \{y_1, \ldots, y_n\}$ of classes. However, in several application scenarios, such as e-discovery (Degnan, 2011, Grossman & Cormack, 2011, Oard & Webber, 2013, Roitblat et al., 2010), online content monitoring (Yang et al., 2021a), and the production of systematic reviews (Callaghan & Müller-Hansen, 2020, Lease et al., 2016, O'Mara-Eves et al., 2015), the amount of documents that must be reviewed may be extremely large; as a result, a number of computerized, *technology-assisted review* (TAR) methods (Grossman & Cormack, 2011, Kanoulas et al., 2019, Yang et al., 2021b), most of them based on some form of supervised learning, have emerged whose goal is to help maximise the cost-effectiveness of the annotators' work.

In e-discovery, an important part of the civil litigation process in many countries (including the US, China, Canada, India, and others), each textual document belonging to a (usually very large) *pool* $\mathcal{P}$ of documents needs to be reviewed for "responsiveness", i.e., the reviewers must decide if it is relevant or not to a topic of interest; in this case, the set of classes of interest is thus $\mathcal{Y} = \{y_r, \overline{y}_r\}$, with y_r the class of responsive documents and $\overline{y}_r$ its complement. E-discovery, though, is usually a *two-stage* review setting since, after review for responsiveness has been carried out, a document deemed responsive needs to be further reviewed for *privilege*, i.e., the reviewers must decide if it contains sensitive material; in this case, the set of classes of interest is thus $\mathcal{Y} = \{y_p, \overline{y}_p\}$, with y_p the class of privileged documents and $\overline{y}_p$ its complement. Documents that are in both y_r and $\overline{y}_p$ are placed in the class y_P of documents that need to be $\underline{\text{p}}$roduced to the other party involved in the litigation; documents that are in both y_r and y_p are $\underline{\text{l}}$ogged (and thus placed in class y_L) into a so-called "privilege log", while documents that are in $\overline{y}_r$ are $\underline{\text{w}}$ithheld from production (and thus placed in class y_W).

In order to reduce the workload of the annotators, a TAR system often uses (for both responsiveness and privilege) some form of supervised learning, which involves the manual annotation of some of the documents in $\mathcal{P}$, training a classifier h by using the manually annotated

* Corresponding author.
E-mail addresses: alessio.molinari@isti.cnr.it (A. Molinari), andrea.esuli@isti.cnr.it (A. Esuli), fabrizio.sebastiani@isti.cnr.it (F. Sebastiani).

https://doi.org/10.1016/j.iswa.2023.200209
Received 3 July 2022; Received in revised form 15 January 2023; Accepted 24 February 2023
Available online 17 March 2023

documents as training examples, and automatically classifying the other documents in $\mathcal{P}$ (or ranking them in terms of estimated degree of membership in the class of interest).[1]

The task of a TAR system is to tell the annotators which documents they should manually review and which other documents they should not, since manually reviewing all documents in $\mathcal{P}$ would be too expensive and would defy the purpose of deploying a TAR system. In doing this, a TAR system should pursue the goal of "getting the job done" (i.e., making sure that the reviewers perform the review process to the satisfaction of the parties involved in the litigation) while at the same time minimising the expected *cost* that the party carrying out the review must incur. There are two types of cost involved in the process: (a) the cost of annotating the documents by responsiveness and by privilege, and (b) the cost that might derive from performing the review process inaccurately (e.g., by unnecessarily producing sensitive material to the other party, or by failing to produce to the other party the necessary material, which might result in a sanction). As far as we know, the only TAR method that explicitly models both types of costs in order to jointly minimise the overall expected cost (i.e., the overall *risk*) of the process is the MINECORE algorithm presented in Oard et al. (2018). This paper adopts the risk minimization approach proposed in Oard et al. (2018), and attempts to explore avenues that allow the performance of MINECORE to be improved.

The rest of this paper is organized as follows. After giving a brief introduction to MINECORE in Section 2, in Sections 3.1 and 3.2 we discuss the two research questions that we tackle in this paper, and that both concern potential ways of improving the performance of MINECORE with respect to the basic setup of the method as presented in Oard et al. (2018). In Section 4 we describe the experimental setting that we have used in order to answer these two questions, while in Section 5 we present and discuss the results of these experiments. Section 6 presents related work on active learning in TAR. Section 7 draws some conclusions and points at avenues for further research. The code and data for reproducing all our experiments is available at https://github.com/levnikmyskin/improved_risk_min_tar.

2. An overview of the MINECORE framework

MINECORE (Oard et al., 2018) is a recently proposed decision-theoretic algorithm for technology-assisted review that attempts to minimise the expected cost (i.e., the risk) of review for responsiveness and privilege in e-discovery.

Given a set $\mathcal{P}$ (the *pool*) of documents that must each be assigned to a class in $\{y_P, y_L, y_W\}$ (where the meaning of these three classes is as discussed in the introduction), the goal of MINECORE is to determine, for each document $\mathbf{x} \in \mathcal{P}$, whether manually reviewing $\mathbf{x}$ for responsiveness and/or privilege is expected to be cost-effective or not. This determination is based

1. on the ("posterior") probabilities of class membership (written as $\Pr(y_r|\mathbf{x})$ and $\Pr(y_p|\mathbf{x})$, hereafter called the *posteriors*) returned by automated classifiers h_r (that classifies documents by responsiveness) and h_p (that classifies documents by privilege);
2. on the costs of manually reviewing a document for responsiveness (λ_r^a) or for privilege (λ_p^a), where superscript a stands for "annotation";
3. on the costs λ_{ij}^m incurred when assigning class y_i to a document which should be assigned class y_j, where $y_i, y_j \in \{y_P, y_L, y_W\}$ and superscript m stands for "misclassification".

Concerning Bullet 2, the fact that costs λ_r^a and λ_p^a are different is due to the fact that annotation by responsiveness can usually be delegated to junior personnel, while annotation by privilege requires more subtle expertise, and is usually entrusted to senior lawyers. Concerning Bullet 3, the fact that costs λ_{ij}^m are different for different $y_i, y_j \in \{y_P, y_L, y_W\}$ is due to the fact that, in e-discovery, not all misclassifications are equally serious; for instance, inadvertently disclosing a privileged document to the other party is typically a very serious mistake, while inadvertently disclosing a nonresponsive nonprivileged document is usually a less serious one.

We assume that our pool $\mathcal{P}$ is partitioned into a set $\mathcal{L}$ of labelled (i.e., manually reviewed for both responsiveness and privilege) documents and a set $\mathcal{U}$ of unlabelled documents.[2]

The MINECORE workflow is articulated in three steps, which we summarise below. In **Step 1** we train from $\mathcal{L}$ the two classifiers h_r and h_p described in Bullet 1 above, and use them to generate, for each document $\mathbf{x} \in \mathcal{U}$, the two posteriors $\Pr(y_r|\mathbf{x})$ and $\Pr(y_p|\mathbf{x})$ mentioned in Bullet 1. We can reasonably assume y_r and y_p to be stochastically independent, which implies that we may assume $\Pr(y_P|\mathbf{x}) = \Pr(y_r|\mathbf{x})\Pr(\overline{y}_p|\mathbf{x})$, $\Pr(y_L|\mathbf{x}) = \Pr(y_r|\mathbf{x})\Pr(y_p|\mathbf{x})$, and $\Pr(y_W|\mathbf{x}) = \Pr(\overline{y}_r|\mathbf{x})$. MINECORE takes a *risk minimization* approach, i.e., it assigns each document $\mathbf{x} \in \mathcal{U}$ to the class

$$\begin{aligned} h(\mathbf{x}) &= \arg \min_{y_i \in \{y_P, y_L, y_W\}} R(\mathbf{x}, y_i) \\ &= \arg \min_{y_i \in \{y_P, y_L, y_W\}} \sum_{j \in \{P,L,W\}} \lambda_{ij}^m \Pr(y_j|\mathbf{x}) \end{aligned} \tag{1}$$

where $R(\mathbf{x}, y_i)$ is the *risk* associated with assigning $\mathbf{x}$ to class $y_i \in \{y_P, y_L, y_W\}$. In other words, MINECORE assigns to each document $\mathbf{x}$ the class that brings about the minimum misclassification risk, thus avoiding assignments which would bring about a high expected misclassification cost. The function for measuring the global misclassification cost (that derives from an assignment of labels in $\{y_P, y_L, y_W\}$ to the documents in $\mathcal{U}$) is thus

$$\begin{aligned} K^m(\mathcal{U}) &= \sum_{\mathbf{x} \in \mathcal{U}} K^m(\mathbf{x}) \\ &= \sum_{\mathbf{x} \in \mathcal{U}} \sum_{i,j \in \{P,L,W\}} \lambda_{ij}^m \cdot \mathbf{1}[\mathbf{x} \in U_{ij}] \end{aligned} \tag{2}$$

where $\mathbf{1}[\cdot]$ is the characteristic function that returns 1 if its argument is true and 0 if it is false, and U_{ij} is the set of documents $\mathbf{x} \in \mathcal{U}$ that are assigned to y_i and whose true class (which we denote by $t(\mathbf{x})$) is y_j. Note that $K^m(\mathbf{x})$ is the misclassification cost brought about by document $\mathbf{x}$, and that the global misclassification cost is simply the sum of document-wise misclassification costs, i.e., MINECORE assumes that misclassification costs are *linear*.[3]

Step 2 is based on the consideration that, if τ_r documents are manually reviewed for responsiveness and τ_p documents are manually reviewed for privilege, the overall cost $K^o(\mathcal{U})$ of the entire process is

[1] In this paper we will assume that the task of a TAR system is to help maximise the cost-effectiveness of the annotators' work when reviewing for responsiveness *and* when reviewing for privilege. In practice, many TAR systems only deal with review for responsiveness, based on the assumption that review for privilege is too important a task to be left to a machine. Our work, and the work of Oard et al. (2018), tries instead to lay the foundations for addressing by automatic means also review for privilege, a task whose automation has received scarce attention in the literature so far (Chhatwal et al., 2020, Vinjumur, 2018, Zhao et al., 2021).

[2] In this paper we will assume, for simplicity, that the same subset of documents $\mathcal{L} \subset \mathcal{P}$ is used for training both h_r and h_p. In reality this need not be the case, and two different subsets $\mathcal{L}_r, \mathcal{L}_p \subset \mathcal{P}$ can be used within MINECORE; this would give rise to two different subsets $\mathcal{U}_r \equiv \mathcal{P} \setminus \mathcal{L}_r$ and $\mathcal{U}_p \equiv \mathcal{P} \setminus \mathcal{L}_p$ of unlabelled documents.

[3] This simplifying assumption is probably a limitation, since in many e-discovery contexts a few mistakes of a certain kind might be without any consequence while more mistakes of the same kind might give rise to major negative consequences, with the relationship between number of mistakes of this type and consequences of these mistakes not being linear. However, dealing with this problem is not within the scope of the present paper, and is a potential topic of future research.

$$\begin{aligned} K^o(\mathcal{U}) &= K^m(\mathcal{U}) + K^a(\mathcal{U}) \\ &= \sum_{\mathbf{x}\in\mathcal{U}} K^m(\mathbf{x}) + \sum_{\mathbf{x}\in\mathcal{U}} K^a(\mathbf{x}) \\ &= \sum_{\mathbf{x}\in\mathcal{U}} K^m(\mathbf{x}) + \lambda^a_r\tau_r + \lambda^a_p\tau_p \end{aligned} \tag{3}$$

where by $K^a(\mathcal{U})$ we indicate the global annotation cost. Similarly to the above, note that $K^a(\mathbf{x})$ is the annotation cost brought about by document $\mathbf{x}$, and that the global annotation cost is simply the sum of document-wise annotation costs, i.e., MINECORE is based on linear annotation costs. Since both misclassification costs and annotation costs are linear, overall costs are also linear, i.e.,

$$K^o(\mathcal{U}) = \sum_{\mathbf{x}\in\mathcal{U}} K^o(\mathbf{x}) \tag{4}$$

If document $\mathbf{x} \in \mathcal{U}$ is reviewed for, say, responsiveness, this has the effect of removing (assuming infallible reviewers) any uncertainty about whether $\mathbf{x}$ is responsive or not. In other words, if by subscript $(n) \in \{(1),(2),(3)\}$ we indicate the value of a given quantity after Step n has been carried out (so that, e.g., $h_{(2)}$ and $\Pr_{(2)}(y|\mathbf{x})$ will indicate the classifier h and the posterior $\Pr(y|\mathbf{x})$ resulting from the completion of Step 2), reviewing $\mathbf{x}$ for responsiveness during Step 2 means that $\Pr_{(2)}(y_r|\mathbf{x})$ will be either 0 or 1. As a result, if during Step 2 document $\mathbf{x} \in \mathcal{U}$ is reviewed for responsiveness, it will in general hold that $\Pr_{(1)}(y_r|\mathbf{x}) \neq \Pr_{(2)}(y_r|\mathbf{x})$, $h_{(1)}(\mathbf{x}) \neq h_{(2)}(\mathbf{x})$ (where h is the cost-sensitive classifier of Equation (1)), and $K^m_{(1)}(\mathbf{x}) \geq K^m_{(2)}(\mathbf{x})$. Since reviewing $\mathbf{x}$ for responsiveness brings about an annotation cost λ^a_r, it is worthwhile to annotate $\mathbf{x}$ only if, as a result of the annotation, $K^o_{(2)}(\mathbf{x}) \leq K^o_{(1)}(\mathbf{x})$, i.e., $K^m_{(2)}(\mathbf{x}) + \lambda^a_r \leq K^m_{(1)}(\mathbf{x})$; in other words, the additional annotation cost λ^a_r must be offset by a reduction $(K^m_{(1)}(\mathbf{x}) - K^m_{(2)}(\mathbf{x}))$ in misclassification cost of greater or equal magnitude. Of course, computing precisely whether annotating $\mathbf{x}$ by responsiveness is going to bring about such a reduction is not possible, because at the time of deciding whether $\mathbf{x}$ should be annotated by responsiveness or not we do not know the value of $y_r(\mathbf{x})$ (a binary variable that indicates whether the reviewers will annotate $\mathbf{x}$ as responsive or not), and we do not know the true label $t(\mathbf{x})$ of $\mathbf{x}$. However, it is possible (see Oard et al., 2018, §3) to compute an expectation of this reduction over the $y_r(\mathbf{x})$ and $t(\mathbf{x})$ variables; when this expected value exceeds λ^a_r, MINECORE decides that $\mathbf{x}$ should be annotated by responsiveness. Since MINECORE computes the expectation of this reduction for all documents in $\mathcal{U}$, this means that MINECORE

- can *rank* the documents in $\mathcal{U}$ (where the top-ranked document is the one with the highest expected reduction), so that by proceeding from the top downwards the annotators first review (by responsiveness) the documents whose annotation again, by responsiveness) brings about the highest expected benefit;
- provides the annotators with a *stopping criterion*, which coincides with the position in the ranked list when the reduction $(K^m_{(1)}(\mathbf{x}) - K^m_{(2)}(\mathbf{x}))$ (actually: its expected value) has become smaller than λ^a_r.

We refer the reader to Oard et al. (2018, §3) for details on how the above expected value is computed, and for a full mathematical specification of Step 2.

Step 3 is essentially identical to Step 2, the only difference being that, while Step 2 focuses on responsiveness, Step 3 focuses on privilege and uses the posteriors $\Pr_{(2)}(y_r|\mathbf{x})$ resulting from Step 2. Note that responsiveness is tackled first because we assume that $\lambda^a_r < \lambda^a_p$; should it be the case that $\lambda^a_r > \lambda^a_p$, MINECORE would deal with privilege in Step 2 and with responsiveness in Step 3.

This concludes the MINECORE workflow. Note that MINECORE enforces a so-called *two-phase TAR workflow* (see e.g. Yang et al., 2021c, §2), i.e., a workflow that consists of

1. a phase in which reviewers annotate, for responsiveness and for privilege, a subset of documents $\mathcal{L} \subset \mathcal{P}$ that are then used to train a classifier for responsiveness h_r and a classifier for privilege h_p, and
2. a phase in which, after the two classifiers have automatically classified all the documents in $\mathcal{U} \equiv \mathcal{P} \setminus \mathcal{L}$, the reviewers annotate some of the latter documents with the goal of correcting possible erroneous labels attributed by the classifiers.

Other TAR systems enforce instead a *one-phase TAR workflow*, where annotators are led to annotate / identify as many responsive / nonprivileged documents (i.e., documents that are candidates for production to the other party) as possible by a process in which a classifier is repeatedly trained, using the documents thus annotated, via some active learning mechanism (typically: using relevance sampling (Lewis & Gale, 1994), a.k.a. "continuous active learning" (Cormack & Grossman, 2015a)). The process is thus akin to searching relevant documents (as in information retrieval) by repeatedly using *relevance feedback* (Rocchio, 1971).

One further difference between MINECORE and most one-phase TAR systems is that, in the latter, only documents that have been manually annotated are produced to the other party; for this reason, these systems lead the annotators to identify as many relevant (i.e., responsive and nonprivileged) documents as early as possible. In MINECORE, instead, the documents that are produced to the other party may or may not have been manually annotated; unlike the above systems, MINECORE leads the annotators to annotate as early as possible the documents that, if not manually annotated, bring about the highest expected misclassification cost.

3. Research questions

We next formulate our two research questions. Interestingly enough, *both questions have to do with how to improve Step 1*, i.e., how to generate the input to Step 2 in such a way that the entire process is most risk-effective.

3.1. Research question # 1: how should we label the training set?

The MINECORE experiments that were presented in Oard et al. (2018) used *passive learning* (PL), i.e., the labelled set $\mathcal{L}$ that was used for training the classifiers h_r and h_p was (conceptually) a random sample of the pool $\mathcal{U}$. This is somehow at odds with the standard practice of the TAR field, according to which the labelled set $\mathcal{L}$ is usually annotated via *active learning* (AL) (Settles, 2012). Active learning is a class of techniques whereby the machine takes an active role in choosing the examples that should be part of the training set $\mathcal{L}$; the most popular such class (and the only class we will consider here) is that of *pool-based* AL techniques, whereby the machine chooses the examples from an available pool $\mathcal{P}$ of unlabelled examples (rather than, say, generating artificial examples with pre-specified properties) and asks the annotators to label them. Here we will consider two popular forms of pool-based AL, *active learning via uncertainty sampling* (ALvUS) (Lewis & Gale, 1994) and *active learning via relevance sampling* (ALvRS) (Lewis & Gale, 1994); we describe both more in detail in Section 4.3, along with a new ALvRUS (*active learning via relevance/uncertainty sampling*) policy, which, as its name suggests, tries to strike a balance between uncertainty sampling and relevance sampling. We choose ALvUS because it is probably the most widely used AL technique in machine learning at large, while we choose ALvRS (which is essentially a form of relevance feedback) because it is probably the most widely used AL technique in TAR, under the name of *continuous active learning* (Cormack & Grossman, 2015a).

In this paper we want to answer the following research question:

RQ1: Assuming we use MINECORE as the TAR system, how should we annotate the training set $\mathcal{L}$? Is it advantageous to

annotate it via passive learning or via an active learning policy?

The question is non-trivial since the three policies (a) typically give rise to classifiers that, for the same number $|\mathcal{L}|$ of training examples, are characterised by different levels of accuracy, and (b) typically give rise to different sets $\mathcal{U} \equiv \mathcal{P} \setminus \mathcal{L}$ of unlabelled sets that the probabilistic classifiers need to rank. This means that one policy may generate a better training set $\mathcal{L}$ (i.e., one that delivers a better classifier) but also generate a set $\mathcal{U} \equiv \mathcal{P} \setminus \mathcal{L}$ harder to rank accurately, and the interactions between these two factors are difficult to predict.

Note that past results showing that AL delivers more accurate classifiers than PL are usually based on testing the two techniques *on the same test set* $\mathcal{U}$. This is not representative of our scenario, where the set $\mathcal{U}$ that the classifiers need to classify changes when the training set $\mathcal{L}$ changes, because $\mathcal{U} \equiv \mathcal{P} \setminus \mathcal{L}$. More in detail, we may expect the ALvUS policy to produce, as the AL literature suggests, the best classifiers (see e.g., Esuli et al., 2019). However, we may expect the PL policy to generate the easiest unlabelled set $\mathcal{U}$ to rank, since when using PL the sets $\mathcal{L}$ and $\mathcal{U}$ are independently and identically distributed (IID), which is not the case when using US or RS. Concerning the RS policy, we may expect it to lead to a high number of relevant (i.e., responsive / privileged) documents being manually (i.e., correctly) labelled, which helps increase recall (and high recall is an important goal in TAR); however, the RS policy is usually the worst of the three in terms of deviation from the IID condition, which means that we may expect it to generate a set $\mathcal{U}$ of documents which is very hard to rank.

When the training set $\mathcal{L}$ has been obtained from $\mathcal{P}$ via active learning, the fact that $\mathcal{L}$ and $\mathcal{U}$ may not be IID is usually true, since in these cases $\mathcal{L}$ is anything but a random sample of $\mathcal{P}$ (i.e., it is a biased sample, suffering from *sample selection bias*), which means that $\mathcal{U} \equiv \mathcal{P} \setminus \mathcal{L}$ is also not a random sample of $\mathcal{P}$. As a consequence, the relationship between $\mathcal{L}$ and $\mathcal{U}$ is one of *dataset shift* (Moreno-Torres et al., 2012, Quiñonero-Candela et al., 2009), i.e., one in which the joint distribution $p_{\mathcal{L}}(\mathbf{x}, y)$ in the labelled data is different from the joint distribution $p_{\mathcal{U}}(\mathbf{x}, y)$ in the unlabelled data. In turn, this means that a classifier trained on $\mathcal{L}$ may perform suboptimally on $\mathcal{U}$.

All in all, this says that it is not easy to predict which of the three methodologies is the most beneficial for MINECORE, and that our research question is an interesting one.

3.2. Research question # 2: should we try to improve the posteriors via SLD?

MINECORE receives as input the posteriors $\Pr(y_r|\mathbf{x})$ and $\Pr(y_p|\mathbf{x})$ returned by the two probabilistic classifiers h_r and h_p. The quality of these posteriors is thus of key importance for the performance of MINECORE. In order to ensure this quality, Oard et al. (2018) subject h_r and h_p to a *calibration* step so that they return well-calibrated probabilities.[4] Calibration routines use k-fold cross-validation, i.e., they tune the classifier in such a way that, when classifying the training documents $\mathbf{x} \in \mathcal{L}$, the classifier returns posterior probabilities $\Pr(y|\mathbf{x})$ that are well-calibrated. In reality, what we are really interested in is that the posteriors of the *unlabelled* documents in $\mathcal{U}$, and not those of the labelled documents in $\mathcal{L}$, are calibrated; if the sets $\mathcal{L}$ and $\mathcal{U}$ are IID, though, if the posteriors of the documents in $\mathcal{L}$ are calibrated, those of the documents in $\mathcal{U}$ are too.

Indeed, virtually all probability calibration routines, including the one used in Oard et al. (2018), assume that $\mathcal{L}$ and $\mathcal{U}$ are IID. If $\mathcal{L}$ and $\mathcal{U}$ are not IID, though, the fact that the posteriors of the documents in $\mathcal{U}$ are well-calibrated is not guaranteed even after the intervention of these routines; in particular, if $\mathcal{L}$ and $\mathcal{U}$ suffer from *prior probability shift* (PPS – (Storkey, 2009)), a form of dataset shift characterized by the fact that the *class prevalence values* (a.k.a., "class relative frequencies", or "class prior probabilities", or simply "priors") $\Pr_{\mathcal{L}}(y)$ and $\Pr_{\mathcal{U}}(y)$ are substantially different for the classes y of interest, the posteriors of the documents in $\mathcal{U}$ will not be calibrated.[5] ALvUS and (especially) ALvRS generate substantial PPS (Settles, 2012)[6]; this means that, if we had to use one of them for generating our training sets $\mathcal{L}$, the quality of the posteriors we would obtain from the resulting classifiers would be a concern.

The Saerens-Latinne-Decaestecker algorithm (hereafter: SLD) (Saerens et al., 2002) is a well-known algorithm (and the only known algorithm) that attempts to improve the quality of the posteriors of the unlabelled documents returned by calibrated probabilistic classifiers in scenarios characterised by PPS. SLD is an iterative algorithm that receives as input the posteriors $\Pr(y|\mathbf{x})$ of the documents $\mathbf{x} \in \mathcal{U}$ returned by the probabilistic classifier. If $\mathcal{L}$ and $\mathcal{U}$ are affected by PPS, it is likely the case that

$$\frac{1}{|\mathcal{U}|} \sum_{\mathbf{x} \in \mathcal{U}} \Pr(y|\mathbf{x}) = \Pr_{\mathcal{U}}(y) \tag{6}$$

does not hold. However, it follows from Equation (5) that Equation (6) is a necessary (though not sufficient) condition (see Esuli et al., 2021, Appendix A) for the posteriors of the documents $\mathbf{x} \in \mathcal{U}$ to be calibrated. SLD is an algorithm that iteratively (i) updates the posteriors $\Pr(y|\mathbf{x})$, and (ii) re-estimates the priors $\Pr_{\mathcal{U}}(y)$, until convergence, i.e., until Equation (6) holds. In other words, while it is not true that SLD calibrates *tout court* the posteriors of the documents belonging to a set $\mathcal{U}$ affected by PPS (because Equation (6) is a necessary but not a sufficient condition for these posteriors to be calibrated), it is true that SLD goes a step in that direction. See Appendix A for a detailed presentation of SLD.

A recent study (Esuli et al., 2021) has shown that, in scenarios characterised by PPS, SLD may be very effective at improving the quality of the posteriors generated by binary classifiers. This would suggest using SLD to improve the quality of the posteriors that are to be fed to MINECORE. However, the experimentation of Esuli et al. (2021) never used active learning for generating the training set $\mathcal{L}$; this means that it is not clear how SLD might perform in our scenario if we had to use either ALvUS or ALvRS.

In this paper we thus want to answer the following research question:

RQ2: Assuming we use MINECORE as the TAR system, can we obtain benefits from using SLD for updating the posterior probabilities that MINECORE receives as input from the two probabilistic classifiers?

[4] We say (see e.g., Flach, 2017) that the posteriors $\Pr(y|\mathbf{x})$, where $\mathbf{x}$ belongs to a set $\mathcal{U}$, are (perfectly) *calibrated* (i.e., accurate) when, for all $a \in [0, 1]$, it holds that

$$\frac{|\{\mathbf{x} \in \mathcal{U} \cap y \mid \Pr(y|\mathbf{x}) = a\}|}{|\{\mathbf{x} \in \mathcal{U} \mid \Pr(y|\mathbf{x}) = a\}|} = a \tag{5}$$

Perfect calibration is usually unattainable for any non-trivial dataset $\mathcal{U}$; however, calibration comes in degrees (and the quality of calibration can indeed be measured), so the goal of classifier calibration routines is to obtain classifiers that return posteriors which are as close as possible to their perfectly calibrated counterparts.

[5] This is a direct consequence of Equation (5).

[6] The reason why ALvUS generates PPS is that it encourages the annotators to annotate documents that are close to the decision threshold; when the dataset is imbalanced, this means that the annotators end up annotating a fairly large proportion of members of the minority class, which means that the prevalence of the minority class in $\mathcal{L}$ ends up being larger than its prevalence in $\mathcal{U}$. Instead, the reason why ALvRS generates PPS is that it encourages the annotators to annotate documents belonging to the minority class; this means that the prevalence of the minority class in $\mathcal{L}$ ends up being *much* larger that its prevalence in $\mathcal{U}$.

The above question may have different answers depending on the answer to RQ1, i.e., depending on whether we annotate the labelled set $\mathcal{L}$ via PL, via ALvUS, via ALvRS, or via ALvRUS, because PL generates no PPS while ALvUS and (especially) ALvRS and ALvRUS generate high PPS.

4. Experiments

4.1. The dataset

There are no publicly available e-discovery datasets in which the documents are annotated by both responsiveness and privilege. As a result, and following (Oard et al., 2018), we use non- e-discovery data that simulate the above conditions. We stick to the original setup used in Oard et al. (2018) and run all of our experiments on the RCV1-v2 dataset, a standard, publicly available benchmark for text classification first presented in Lewis et al. (2004) and consisting of 804,414 news stories produced by Reuters from 20 Aug 1996 to 19 Aug 1997.[7] RCV1-v2 ranks as one of the largest corpora currently used in text classification research. RCV1-v2 is multi-label, i.e., a document may belong to several classes at the same time. In Lewis et al. (2004) the collection is partitioned into a training set of 23,149 documents and a test set of 781,265 documents. For computational reasons, for our experiments we only use the first 100,000 documents of the collection, which include the 23,149 documents used for training in Lewis et al. (2004).

In Oard et al. (2018), RCV1-v2 classes were chosen to simulate the classes of responsive documents (y_r) and privileged documents (y_p), respectively. More specifically, the authors of Oard et al. (2018) chose all pairs (y_r, y_p) of RCV1-v2 classes such that the class prevalence value $\Pr_{\mathcal{P}}(y_r)$ of y_r is in the [0.03,0.07] interval and the class prevalence value $\Pr_{\mathcal{P}}(y_p|y_r)$ of y_p in the set of responsive documents y_r is in [0.01,0.20], where class prevalence values are computed on the entire pool $\mathcal{P}$. This broad range is actually seen in e-discovery practice, with some classification tasks run on very imbalanced data, and others run on data that have been prescreened at acquisition time to have as high a prevalence value for the responsive class as can be achieved (Oard & Webber, 2013). For each of the 24 RCV1-v2 classes that meet the class prevalence value criterion for simulating y_r, the authors of Oard et al. (2018) randomly selected 5 classes that meet the class prevalence value criterion for simulating y_p: this gives rise to 120 class pairs. The experiments described in this paper were run on the very same set of class pairs as used in Oard et al. (2018).

4.2. The evaluation measure

In order to evaluate the performance of MINECORE under the different setups, we employ the same *cost structures* (i.e., sets of actual values for the λ^a_r, λ^a_p, and λ^m_{ij} costs that MINECORE relies on) $\Lambda = (\Lambda^a, \Lambda^m)$ as used in Oard et al. (2018); the three cost structures were originally elicited from three different e-discovery experts. We employ the original notation as used in Oard et al. (2018), where:

- $\Lambda^a = (\lambda^a_r, \lambda^a_p)$, where λ^a_r denotes the cost of annotating a single document for responsiveness and λ^a_p denotes the cost of annotating a single document for privilege, and where the a superscript stands for "annotation";
- $\Lambda^m = \{\lambda^m_{ij}\}$ (for $i, j \in \{P, L, W\}$), where each entry λ^m_{ij} represents the cost incurred when misclassifying an element of y_j into y_i (the m superscript stands for "misclassification").

[7] http://trec.nist.gov/data/reuters/reuters.html.

Table 1
Cost structures used in our experiments, as elicited by the authors of Oard et al. (2018). Each individual cost is expressed in USD.

Λ	λ^a_r	λ^a_p	λ^m_{PL}	λ^m_{PW}	λ^m_{LP}	λ^m_{LW}	λ^m_{WP}	λ^m_{WL}
Λ_1	1.00	5.00	600.00	5.00	150.00	3.00	15.00	15.00
Λ_2	1.00	5.00	100.00	0.03	10.00	2.00	8.00	8.00
Λ_3	1.00	5.00	1000.00	0.10	1.00	1.00	1.00	1.00

The three cost structures Λ_1, Λ_2, and Λ_3 are displayed in Table 1.[8] Given a cost structure, we compute the overall cost of a given policy by means of Equation (3), and use overall cost as our evaluation function.

As we have previously remarked, every sampling policy used for active learning (as well as the random sampling policy used for passive learning) generates a different set $\mathcal{L}$ of annotated documents, and (as a consequence) a different set $\mathcal{U} \equiv \mathcal{P} \setminus \mathcal{L}$ of unlabelled documents. We evaluate MINECORE on $\mathcal{U}$ only (since overall cost on $\mathcal{L}$ is the same for every tested policy),[9] even if $\mathcal{U}$ is different for every policy we consider; this is justified by the fact that a given policy has to be evaluated also in terms of how hard to classify is the set $\mathcal{U}$ it generates.

4.3. The active learning methods

In our experiments we test MINECORE on training sets $\mathcal{L}$ generated by three active learning policies, i.e., *Active Learning via Uncertainty Sampling* (ALvUS), *Active Learning via Relevance Sampling* (ALvRS), and *Active Learning via Relevance/Uncertainty Sampling* (ALvRUS). Other, more recent active learning algorithms such as those presented by Huang et al. (2014) and Dasgupta and Hsu (2008) were initially explored but finally dismissed due to their unfeasibly expensive computational cost (see Section 4.6 for more on this). We here briefly describe the three active learning policies we use.

Active Learning via Uncertainty Sampling (ALvUS). In ALvUS, an interactive process asks the reviewers to annotate for responsiveness/privilege the b documents in $\mathcal{U}$ for which $\Pr(y_i|\mathbf{x})$ (with $i \in \{r, p\}$) is closest to 0.5 (parameter b is known as the *batch size*). The set of probabilities $\Pr(y_i|\mathbf{x})$, for all $\mathbf{x} \in \mathcal{U}$, is obtained from a classifier h_i trained (and calibrated) on an initial training set $\mathcal{L}$, with $\mathcal{P} \equiv \mathcal{L} \cup \mathcal{U}$. The set of b documents that the annotators then label is added to the training set $\mathcal{L}$ and removed from the unlabelled set $\mathcal{U}$, h_i is retrained, and $\mathcal{U}$ is classified for responsiveness/privilege again; this process is repeated until we have annotated a predefined number n of documents.

Active Learning via Relevance Sampling (ALvRS). A variant of the previous policy is obtained when the active learning process asks the reviewers to annotate for responsiveness/privilege the b documents in $\mathcal{U}$ for which $\Pr(y_i|\mathbf{x})$ (with $i \in \{r, p\}$) *is highest*. The rest of the process is as in ALvUS. Both ALvUS and ALvRS were originally introduced in Lewis and Gale (1994).

Active Learning via Relevance/Uncertainty Sampling (ALvRUS). A variant of the two previous policies consists of asking the reviewers to

[8] These cost structures indicate that the three experts from which they were elicited have very different beliefs about misclassification costs, possibly deriving from having different use cases in mind. The 3rd expert (the one who returned cost structure Λ_3) holds the most "extreme" views, since s/he believes that what matters overwhelmingly more than anything else is that no privileged documents are inadvertently produced to the other part ($\lambda_{PL} = 1000.00$), and that no other mistake matters much. The 1st expert is also fairly extreme, in that s/he also believes that placing on the privilege log a document that should instead have been produced also has an extremely high cost ($\lambda_{LP} = 150.00$). The least extreme expert seems to be the 2nd one.

[9] This derives from (i) the fact that annotating $|\mathcal{L}|$ documents has, under our assumption that all documents have an annotation cost of λ^a_r or λ^a_p, the same cost irrespectively of the policy used for annotating them, and (ii) the fact that, once annotated, these $|\mathcal{L}|$ documents have zero misclassification cost, irrespectively of the policy used for annotating them.

annotate, at each iteration, the $b/2$ documents in $\mathcal{U}$ for which $\Pr(y_i|\mathbf{x})$ is closest to 0.5 *and* the $b/2$ documents in $\mathcal{U}$ for which $\Pr(y_i|\mathbf{x})$ is highest (with $i \in \{r, p\}$). In other words, this policy is a mix of ALvUS and ALvRS (hence its name), since it attempts to satisfy both goals at the same time, i.e., providing feedback on the regions of the instance space in which the classifier is weakest (as in ALvUS) and adding many examples from the "positive class" (i.e., y_r or y_p) to $\mathcal{L}$ (as in ALvRS).

4.4. Passive learning

By *passive learning* (PL) we mean a simple strategy that generates a training set $\mathcal{L}$ by uniformly sampling from the data pool $\mathcal{P}$. This was the technique used in Oard et al. (2018) and, as such, we use it as a baseline.

4.5. The Rand(RS), Rand(US), and Rand(RUS) policies

We add to our experiments three "oracle-like" policies (*Rand(RS)*, *Rand(US)*, *Rand(RUS)*), which will serve as testing grounds for the other algorithms that we run for answering RQ1 and RQ2. The goal of the Rand(RS) (resp., Rand(US), Rand(RUS)) policy is that of building a training set $\mathcal{L}$ and a test set $\mathcal{U}$, from the same pool $\mathcal{P}$ where the other AL policies operate, by performing a "controlled" random sampling of the documents, i.e., controlled in such a way that it yields the same prevalence value $\Pr_{\mathcal{L}}(y_i)$ of the positive class (with $i \in \{r, p\}$) that we would have obtained had we used ALvRS (resp., ALvUS, ALvRUS).

That is, Rand(RS) (resp., Rand(US), Rand(RUS)) builds a pair consisting of a training set $\mathcal{L}$ and a test set $\mathcal{U}$ such that $\Pr_{\mathcal{L}}(y_i)$ (with $i \in \{r, p\}$) is identical to the prevalence of y_i in the training set built via ALvRS (resp., ALvUS, ALvRUS), even if the selection of the documents for generating $\mathcal{L}$ is random. In this case, it also follows that the prevalence value $\Pr_{\mathcal{U}}(y_i)$ (with $i \in \{r, p\}$) of the positive class in $\mathcal{U}$ is identical to that in the test set built via ALvRS (resp., ALvUS, ALvRUS). In terms of dataset shift, we can say that the three original active learning policies generate shift in the distribution of the class priors (i.e., $p_{\mathcal{L}}(y) \neq p_{\mathcal{U}}(y)$) *and* in the distribution of the covariates conditional on the classes (i.e., $p_{\mathcal{L}}(\mathbf{x}|y) \neq p_{\mathcal{U}}(\mathbf{x}|y)$), while the corresponding Rand policies generate shift of the former type but not of the latter type (i.e., do not suffer from *sampling bias*).

Comparing the three original active learning policies with the corresponding Rand policies should help us understand whether the results we are seeing are due to the prevalence values of the positive class in $\mathcal{L}$ and $\mathcal{U}$ that the AL policies generate, or to the data selection criteria.

4.6. Exploring other policies

Motivated by the advances in the active learning literature, we decided to explore other more recent and sophisticated AL policies. Unfortunately, the ones we considered were computationally expensive or prohibitive to run, and we eventually decided not to use them in our experiments. The following is an overview of the two policies we considered and of the reasons why they proved too challenging to run.

Active Learning by Querying Informative and Representative Examples (QUIRE). QUIRE (Huang et al., 2014) is a recent (with respect to ALvUS and ALvRS) active learning algorithm whose goal is not only to annotate documents for which the classifier exhibits the strongest uncertainty (as in ALvUS), but also to maximise the representativeness (i.e., diversity) of the examples annotated at every iteration. For more technical details we refer the reader to Huang et al. (2014).[10]

However, this technique is problematic since it requires (a) the computation of an $m \times m$ kernel matrix K, where m is the number of documents in pool $\mathcal{P}$, and (b) the storage of another $m \times m$ matrix $L = (K + \lambda I)^{-1}$ (where I is the identity matrix). This is clearly unfeasible in our case, where $m = 100{,}000$ (assuming one byte for each element of each matrix, just storing K and L would require approximately 160GB); note also that, in real e-discovery scenarios, $\mathcal{P}$ may contain many more documents than the 100,000 documents we use in our experiments.

Active Learning via Hierarchical Sampling (ALvHS). ALvHS was first presented in Dasgupta and Hsu (2008). The goal of this algorithm is to avoid sampling bias, i.e., the fact that the set of labelled documents $\mathcal{L}$ may not be representative of the remaining set of unlabelled documents $\mathcal{U}$, which instead happens when using AL strategies such as ALvRS, ALvUS, or ALvRUS. The basic step of this policy consists of partitioning the data into hierarchical clusters and later sampling from them (for a more in-depth presentation of the algorithm see Dasgupta & Hsu, 2008).

However, given m items to be clustered, hierarchical clustering algorithms have a complexity of $\mathcal{O}(m^3)$, which can be reduced to $\mathcal{O}(m^2)$ or $\mathcal{O}(m^2 \log m)$ only in some specific cases (Patel et al., 2015). In any case, the algorithm results in unfeasibly expensive computation costs in our application scenario.

4.7. The experimental setup

In order to answer RQ1, we compare our different active / passive learning policies for training the two classifiers h_r and h_p used in Step 1 of MINECORE.

For these experiments, we take the first 100,000 documents of the RCV1-v2 collection as the pool of documents $\mathcal{P}$ to which MINECORE is applied. As already mentioned in Section 4.1, for higher consistency with the experiments carried out in Oard et al. (2018), we use the same RCV1-v2 pairs of classes used in Oard et al. (2018) to play the role of the responsive class y_r and the privileged class y_p; this is a set of 120 pairs of RCV1-v2 classes (see Section 4.1 for details).

For each active / passive learning policy we test, we run different experiments in which we vary the size s of the training set that the process eventually creates; we test all sizes $s \in \{2000, 4000, 8000, 16000, 23149\}$; the reason why we use the fairly peculiar size $s = 23149$ is that this is the size used in Oard et al. (2018).

We seed all the active learning processes with a set $\mathcal{S}$ of 1000 initial training documents randomly sampled from our pool $\mathcal{P}$, train (for each $y \in \{y_r, y_p\}$) an SVM classifier on $\mathcal{S}$, calibrate it, and apply it to all the remaining unlabelled documents in $\mathcal{U} \equiv \mathcal{P} \setminus \mathcal{S}$ to obtain posterior probabilities $\Pr(y|\mathbf{x})$ for each of them.[11] We constrain this initial training set $\mathcal{S}$ to have at least 2 positive instances, which are the bare minimum in order to calibrate the classifier.

We then iterate the active learning process on the remaining unlabelled documents with a batch size $b = 1000$. The active learning process simulates the work of *infallible* reviewers, i.e., at each iteration the unlabelled documents are ranked based on their assigned posteriors, the b unlabelled documents which are ranked highest are added to the training set together with their true label (which simulates the infallible reviewer's annotation) and removed from the unlabelled set $\mathcal{U}$, the classifier is retrained, recalibrated, and applied again to all the remaining unlabelled documents to obtain updated posterior probabilities for each of them.

As mentioned in Section 4.5, for each training set size $s \in \{2000,$ $4000,$ $8000,$ $16000,$ $23149\}$ we also generate training sets of size s via the $Rand(US)$, $Rand(RS)$, $Rand(RUS)$, and PL policies; for each such policy all sets are obtained each time anew via random sampling (constrained for $Rand(US)$, $Rand(RS)$, and $Rand(RUS)$ – see Section 4.5, unconstrained for PL), i.e., it is *not* the case that, for a given policy, the

[10] An implementation of this algorithm is available at https://libact.readthedocs.io/en/latest/libact.query_strategies.html#module-libact.query_strategies.quire.

[11] Since we use SVMs as the base learner, calibration is strictly necessary, since SVMs return confidence scores that are not probabilities; our calibration step thus maps these confidence scores into calibrated posterior probabilities. In all of our experiments we use Platt's calibration method (Platt, 2000).

smaller training sets are contained in the larger ones. Here too, the classifier is trained, calibrated, and applied to all the remaining unlabelled documents, after which these latter are ranked based on their newly obtained posterior probabilities.

For any of these policies, the finally obtained posterior probabilities for each of the remaining unlabelled documents are fed to Step 2 of the MINECORE workflow. We run Steps 2 and 3 of the MINECORE workflow for any of the above policies and for each cost structure (Table 1), and evaluate MINECORE as explained in Section 4.2, so as to ascertain which policy brings about the smallest overall cost. We will show and comment these results in Section 5.1.

In order to answer RQ2, instead, we run SLD on the posterior probabilities (here indicated as $\Pr^{\mathrm{PreSLD}}(y|\mathbf{x})$) obtained from each of the above policies, thus obtaining $\Pr^{\mathrm{PostSLD}}(y|\mathbf{x})$ posterior probabilities, and we compare, for each policy and each cost structure, the overall cost resulting from using PreSLD posteriors with the overall cost resulting from using, in place of them, PostSLD posteriors. These results are commented on in Section 5.2.

5. Results

In this section we show and analyse our results for RQ1 (Section 5.1) and RQ2 (Section 5.2). For RQ1 we present our results in Tables 3 and 4. For RQ2 we illustrate our results in Tables 5, 6, 8, 9, 10, and 11. Finally, we show some insightful plots concerning the AL policies in Figs. 1 and 2, and other plots concerning the effects of SLD on the distribution of the posterior probabilities in Figs. 3 to 5.

5.1. RQ1

The goal of our first research question (RQ1) is that of understanding which among the policies described in Sections 4.3 to 4.5 generates the best training sets on which to train our two classifiers.

ALvRS (in its CAL incarnation) is the standard active learning methodology used in one-phase TAR systems. In these contexts, we are given an unlabelled pool of documents and our goal is to find the highest number of relevant (i.e., positive class) documents in the least possible amount of time; given this goal, relevance sampling is usually preferred to uncertainty sampling, as it encourages the annotators to review documents that are likely to be relevant. However, MINECORE operates according to a two-phase TAR workflow, where in the 1st phase we create a training set which is later used to train the classifier, and where the posteriors returned by this classifier are then used as input by MINECORE in the 2nd phase in order to prioritise the documents that the annotators should review. That is, in this case ALvRS might be less effective than ALvUS, since, by repeatedly improving the classifier in the regions of instance space on which the classifier is most uncertain, ALvUS might eventually obtain higher-quality posteriors. Moreover, we might expect the ALvRUS policy, which stands as a middle ground between ALvRS and ALvUS, to also improve MINECORE's results by building a higher-quality classifier than ALvRS.

In Table 2 we report the average recall[12] and the average class prevalence in the training set $\mathcal{L}$ and in the set of unlabelled documents $\mathcal{U}$ as deriving from the application of the different AL policies, where the averages are computed across y_r and y_p. All these figures are reported at different training set sizes (i.e., 2000, 4000, 8000, 16000, 23149); this should give a clearer picture of the overall scenarios generated by the different active learning policies.

The results in Table 3 show that, for any given cost structure, ALvRUS is the most effective among the policies we have studied. Regarding the other policies, ALvUS appears to be the second-best policy for two cost structures out of three (Λ_1 and Λ_2), whereas ALvRS

Table 2
Average recall and training/test class prevalence values at different training set sizes for the three active learning policies.

	$\lvert\mathcal{L}\rvert$	Recall	$\Pr_{\mathcal{L}}(y)$	$\Pr_{\mathcal{U}}(y)$
ALvRS	2,000	0.240 ± 0.113	0.698 ± 0.056	0.080 ± 0.099
	4,000	0.509 ± 0.218	0.762 ± 0.113	0.064 ± 0.100
	8,000	0.766 ± 0.231	0.637 ± 0.203	0.045 ± 0.095
	16,000	0.894 ± 0.158	0.427 ± 0.252	0.028 ± 0.077
	23,149	0.936 ± 0.107	0.336 ± 0.254	0.019 ± 0.061
ALvUS	2,000	0.099 ± 0.044	0.300 ± 0.047	0.088 ± 0.099
	4,000	0.196 ± 0.074	0.308 ± 0.068	0.083 ± 0.099
	8,000	0.431 ± 0.161	0.337 ± 0.072	0.071 ± 0.101
	16,000	0.597 ± 0.186	0.251 ± 0.089	0.062 ± 0.102
	23,149	0.670 ± 0.183	0.204 ± 0.091	0.059 ± 0.102
ALvRUS	2,000	0.135 ± 0.060	0.408 ± 0.067	0.086 ± 0.099
	4,000	0.381 ± 0.176	0.560 ± 0.057	0.073 ± 0.100
	8,000	0.699 ± 0.249	0.551 ± 0.129	0.052 ± 0.100
	16,000	0.863 ± 0.197	0.388 ± 0.181	0.036 ± 0.091
	23,149	0.914 ± 0.150	0.307 ± 0.191	0.027 ± 0.081

Table 3
Average overall costs resulting from running MINECORE when different policies have been used for generating the training sets. Values in **boldface** indicate the best results for the given cost structure. The reported values are obtained by averaging across the experiments run with different training set sizes (2000, 4000, 8000, 16000, 23149). Superscripts † and ‡ indicate whether the second-best method is not statistically significantly different from the best one, according to a Wilcoxon signed-rank test at different confidence levels: symbol † indicates $0.001 < p < 0.05$, while ‡ indicates $p \geq 0.05$. In this table, all differences are statistically significant, hence no instances of † and ‡ are present.

Λ	PL	ALvRS	ALvUS	ALvRUS
Λ_1	38,589	39,373	32,511	**19,890**
Λ_2	13,047	10,398	6,721	**5,766**
Λ_3	5,318	2,742	3,749	**2,129**

Table 4
Same as Table 3, but with different policies for generating the training sets.

Λ	Rand(RS)	Rand(US)	Rand(RUS)
Λ_1	63,109	**41,284**	41,298
Λ_2	14,708	**8,397**	9,819
Λ_3	4,102	4,087	**2,964**

achieves second-best results on Λ_3. Finally, the passive learning strategy is the worst one in two cases out of three (Λ_2 and Λ_3).

Furthermore, the Rand(RS), Rand(US), and Rand(RUS) results of Table 4 tell us that a higher prevalence value of the positive class in the test set, and an overall better balance between training and test class prevalence values (which we obtain when using uncertainty sampling), can steer the results in favour of ALvUS if, as with all the Rand policies, $\mathcal{L}$ and $\mathcal{U}$ do not suffer from sampling bias. Also, despite the fact that ALvRUS is the best policy in the results of Table 3, its corresponding Rand policy does not achieve the best results for two cost structures out of three (Λ_1 and Λ_2), albeit still obtaining better results than the *Rand(RS)* policy.

That said, we conclude that *the best strategy for training the h_r and h_p classifiers on which Step 1 of MINECORE hinges is, by a wide margin, ALvRUS*.

We end this discussion by noting that an obvious way to try to improve on ALvRUS would consist in adding to it a parameter α, so that

[12] In this context, by *recall* we mean the ratio $\Pr_{\mathcal{L}}(y)/\Pr_{\mathcal{P}}(y)$ between the number of positive documents in the training set $\mathcal{L}$ and the total number of positive documents in the entire pool $\mathcal{P}$.

Table 5
Average overall costs resulting from running MINECORE on posterior probabilities coming from either the classifier (PreSLD) or the SLD algorithm (PostSLD). Notational conventions are as in Table 3. A value in **boldface** indicates the best result on the given row (i.e., combination of a cost structure and a choice between PreSLD and PostSLD), whereas a value in underline indicates the best result for the given cost structure.

	Λ	PL		ALvRS		ALvUS		ALvRUS	
Λ_1	PreSLD	38,589		39,373		32,511		**19,890**	
	PostSLD	38,620	+0.08%	28,694	-37.22%	**25,309**‡	-28.46%	33,504	+40.63%
Λ_2	PreSLD	13,047		10,398		6,721		**5,766**	
	PostSLD	13,073	+0.20%	14,040	+35.03%	**7,299**	+8.6%	14,612	+153.42%
Λ_3	PreSLD	5,318		2,742		3,749		**2,129**	
	PostSLD	5,317	-0.02%	**3,735**	+36.21	8,452	+125.42%	8,748	+310.90%

Table 6
Same as Table 5, but with different policies for generating the training sets.

	Λ	Rand(RS)		Rand(US)		Rand(RUS)	
Λ_1	PreSLD	63,109		**41,284**		41,298	
	PostSLD	14,394	-77.19%	19,819	-51.99%	**12,311**	-70.19%
Λ_2	PreSLD	14,708		**8,397**		9,819	
	PostSLD	**2,586**	-82.42%	4,641	-44.73%	3,065	-68.79%
Λ_3	PreSLD	4,102		4,087		**2,964**	
	PostSLD	1,736	-57.68%	2,857	-30.10%	**1,586**	-46.49%

the reviewers are asked to annotate, at each iteration, the $\alpha \cdot b$ documents in $\mathcal{U}$ for which $\Pr(y_i|\mathbf{x})$ is closest to 0.5 and the $(1-\alpha)\cdot b$ documents in $\mathcal{U}$ for which $\Pr(y_i|\mathbf{x})$ is highest (with $i \in \{r,p\}$); optimising this parameter (say, on a held-out dataset) would allow striking the best possible balance between "exploration" (the US component) and "exploitation" (the RS component).

5.2. RQ2

The goal of our second research question (RQ2) is that of understanding whether the application of SLD can (i) improve the quality of the posterior probabilities that are input to Step 2 of the MINECORE algorithm, and thus (ii) generate a reduction in overall cost.

The active learning strategies that we use in this paper (and ALvUS, the "winner" in the previous batch of experiments, is no exception) tend to generate PPS between the training set and the set of unlabelled data; more specifically, they tend to generate situations in which the class prevalence value $\Pr_{\mathcal{L}}(y)$ (with $y \in \{y_r, y_p\}$) can be much larger than the class prevalence value $\Pr_{\mathcal{U}}(y)$. In such a scenario, given the findings of Esuli et al. (2021), we would expect the application of SLD to bring about substantial improvements to the quality of the posterior probabilities.

The results of our experiments are displayed in Table 5. Something we can see from these figures is that the results of the application of SLD are uneven; in some cases this application brings about a reduction in overall cost, while in other cases overall cost increases. In particular, SLD always brings about a deterioration when ALvRUS (the "winning" policy of our previous section) has been used to generate the training set. In sum, there is no clear answer to RQ2.

However, it is important to notice that, for all three cost structures, the best result is obtained by using ALvRUS and *not* using SLD. We thus have a clear answer to RQ1 and RQ2 altogether, i.e., that *the best course of action is to avoid using SLD and stick to the "PreSLD" posterior probabilities generated by classifiers trained via ALvRUS*.

In case we wonder what are the reasons for the failure of SLD to systematically improve the quality of our posterior probabilities, the answer comes from examining Table 6, which presents the results of experiments analogous to the ones of Table 5 but using the Rand active learning policies instead of the original ones. Table 6 says that, for all Rand policies and for all cost structures, the SLD algorithm brings about a *drastic* reduction in overall cost, unlike what happened for the original active learning policies. It is thus easy to conclude that *SLD copes well with PPS* (which the Rand policies generate) *but not with sampling bias* (which the original active learning policies, unlike the Rand policies, generate). Indeed, a close examination of SLD (see Algorithm 1 in Appendix A) shows that nothing in it caters for sampling bias. Conversely, SLD does cater for PPS; indeed, if we assumed that there is no PPS between $\mathcal{L}$ and $\mathcal{U}$, there would be no need to re-estimate the priors (see Line 11 of Algorithm 1) and, consequently, to update the posteriors (see Line 13), as SLD instead does.

Incidentally, the fact that Rand(RS) and Rand(RUS) are the two best overall algorithms confirms the observations of Esuli et al. (2021) that the greater the shift between training set and test set, the better the performance of SLD; indeed, Rand(RS) and, to a lesser degree, Rand(RUS), tend to generate more PPS than Rand(US).

In order to provide a finer-grained analysis of the above results, we further

1. Bin the classes by class prevalence value, in order to analyse whether the results may depend on the prevalence value of the class;
2. generate a visualization of the results of the different selection strategies (i.e., ALvRS, ALvUS, ALvRUS, PL and Rand), so as to see where the documents they select are picked from the data distribution;
3. Plot the distributions of the posteriors before SLD and after SLD, in order to visually understand how the SLD algorithm is adjusting these distributions.

5.2.1. Effects of class prevalence value

Regarding Point 1, we bin the 28 classes in quartiles by prevalence value; we accordingly call these quartiles "Low", "Medium-Low", "Medium-High", "High". In Table 7 we show which bin each class belongs to and whether the class is used to simulate Responsiveness (R), Privilege (P), or both (R+P).

We show in Tables 8 and 9 the results for each of the four bins, for responsiveness and privilege, respectively; these results are consistent with those of Table 5, with the PL strategy only slightly negatively af-

Table 7
The RCV1-v2 classes that we use in our experiments, binned into quartiles according to prevalence value. The last column indicates whether we use the class to represent responsiveness (R), privilege (P), or both (R + P). We use these quartiles to bin our results in Tables 8, 9, 10, 11.

Class	Prevalence	Quartile	Used for
C21	0.032	Low	R + P
M12	0.032	Low	R + P
M132	0.033	Low	R + P
E12	0.034	Low	R + P
E212	0.034	Low	R + P
M131	0.035	Low	R + P
C24	0.040	Medium-Low	R + P
GCRIM	0.040	Medium-Low	R + P
GVIO	0.041	Medium-Low	R + P
C13	0.047	Medium-Low	R
GDIP	0.047	Medium-Low	R + P
C31	0.050	Medium-Low	R + P
C17	0.052	Medium-High	R + P
E21	0.054	Medium-High	R + P
C181	0.054	Medium-High	R + P
M141	0.059	Medium-High	R + P
M11	0.061	Medium-High	R + P
C18	0.066	Medium-High	R + P
M13	0.067	High	R + P
GPOL	0.071	High	R + P
C152	0.091	High	R + P
C151	0.102	High	R + P
M14	0.106	High	R + P
ECAT	0.149	High	R + P
C15	0.189	High	P
MCAT	0.255	High	P
GCAT	0.297	High	P
CCAT	0.474	High	P

fected by SLD and the three AL strategies suffering the strongest effects. Notice how the bins of classes with lower prevalence values (Low and Medium-Low) tend to be the ones where SLD performs comparatively better; given that SLD works better in high-shift scenarios (Esuli et al., 2021), this was to be expected, since the AL strategies will cause a much stronger PPS if the overall class prevalence value is low (since the few examples of the positive class tend to end up quickly in the training set). Also, notice how the above-described effect is stronger for ALvRS than for ALvUS; this was also to be expected, as ALvUS generates less extreme PPS than ALvRS. Finally, the results of Table 9 indicate that the deterioration brought about by SLD is higher for the privilege classes than it was for the responsiveness classes (Table 8); again, this was to be expected, since the privilege class has a greater impact on the final cost of a review than the responsiveness class, since $\lambda_p^a > \lambda_r^a$.

As we can see from Tables 10 and 11, the situation is completely reversed when considering the Rand policies: SLD performs consistently well, across all bins and cost structures, be it for responsiveness or privilege.

In conclusion, the results of this analysis confirm that no matter the class prevalence value and the cost structure, SLD tends to have a clear negative effect on the quality of the posteriors generated by classifier trained via active learning techniques.

5.2.2. *Visualising the behaviour of different selection strategies*

Let us now move to Point 2, i.e., visualising the effects of the different selection strategies. As we have mentioned before, the results we have just commented say that the reasons behind the failure of SLD to improve the posteriors have to be found in the document selection criteria enacted by our three original active learning strategies. In order to better understand how these different strategies select their documents from the pool $\mathcal{P}$, we provide a visualization of the first 1000 documents that each policy selects: in order to do this, we remove the initial seed set $\mathcal{S}$, use the LSA algorithm (Landauer et al., 1998) on the TF-IDF matrix to reduce its dimensionality to the two largest singular values, pick two random classes (namely the C17 and M14 classes of RCV1-v2), and show the corresponding scatter plots for the two components in Fig. 1. (For best visualization results use the .pdf version of the present paper and zoom in as necessary.) The plots show, with different colours, the positive and negative documents of $\mathcal{P}$, and the positive and negative documents that the given strategy selected to be included in $\mathcal{L}$.[13]

It is indeed interesting to see how the ALvRS policy selects mostly positive instances, and very few negative ones, and that all of these positive instances tend to come from the same region of the instance space, thereby bringing about a training set characterised by very little diversity; this pattern is especially evident for the C17 class, and is less evident but still present for the M14 class. This is a visualisation of the sampling bias brought about by ALvRS (see also Dasgupta & Hsu, 2008, §2 and Krishnan et al., 2021). The same visualisation also shows how the Rand(RS) policy selects instead the (same amount of) positives in a much more uniformly distributed and unbiased way. Similar effects, although less marked, can be noted for ALvRUS and ALvUS (and for the corresponding Rand policies).

The plots for the passive learning (PL) strategy show instead the intrinsic limitations of this policy in TAR contexts (also highlighted in Cormack and Grossman (2014) for one-phase TAR systems); given the substantial imbalance between the positive and negative examples, a raw random sampling of the data results in the selection of very few positive documents; this is evident for both C17 and M14.

5.2.3. *Analysing the distributions of the posteriors*

Let us now discuss Point 3. Despite the fact that the LSA plots can help us visualise the effects of the different selection strategies, they cannot tell us much about the consequences of each strategy and why these strategies cause SLD to fail in delivering better-quality posterior probabilities. A much more helpful insight might instead emerge by plotting the distributions of the PreSLD and PostSLD posteriors of the unlabelled documents as returned by the classifiers trained via the different selection strategies that we consider. For doing this we pick one of the two classes of our previous example (the C17 class) and plot (see Figs. 3 and 4) the above-mentioned distributions for classifiers trained on 2,000 documents and on 23,149 documents, respectively. We also show the CCAT class, one of the most populated RCV1-v2 classes, for comparison (see Fig. 5).[14] We use a logarithmic scale for the Y axis as this helps to visualise the distributions better.

Looking at Fig. 4 we can observe that the PreSLD distributions for all three AL policies are extremely skewed towards zero (i.e., most of the mass of the probability distribution is close to the zero value), whereas the distributions for the respective Rand policies are more uniformly spread across the [0,1] interval. The hypothesis for the cause of SLD's extremisation of the posteriors distribution might then arise from these plots. The SLD algorithm iteratively updates the posterior and prior probabilities using equations

$$\hat{\Pr}_{\mathcal{U}}^{(s)}(y) = \frac{1}{\mathcal{U}} \sum_{\mathbf{x}\in\mathcal{U}} \overset{(s-1)}{\Pr}(y|\mathbf{x})$$
$$\Pr^{(s)}(y|\mathbf{x}) = \frac{\dfrac{\hat{\Pr}_{\mathcal{U}}^{(s)}(y)}{\Pr_{\mathcal{L}}(y)} \cdot \overset{(0)}{\Pr}(y|\mathbf{x})}{\sum_{y\in\mathcal{Y}} \dfrac{\hat{\Pr}_{\mathcal{U}}^{(s)}(y)}{\Pr_{\mathcal{L}}(y)} \cdot \overset{(0)}{\Pr}(y|\mathbf{x})} \tag{7}$$

[13] Notice that given the scale and the density of the documents, it may look as if some negative selected points are among the positive ones. This is just a limitation of the 2-D projection.

[14] Notice that for the CCAT class we only plot the distributions for the AL classifiers. Distributions for the *Rand* classifiers were fairly similar and thus not very interesting.

Table 8
Average MINECORE overall costs with responsiveness classes binned by prevalence value. A positive increment indicates higher costs resulting from the application of SLD. Superscript † and ‡ denote whether the PostSLD results are not statistically significantly different from the PreSLD results, according to a Wilcoxon signed-rank test at different confidence levels: symbol † indicates $0.001 < p < 0.05$, while ‡ indicates $p \geq 0.05$.

	Bin	PL			ALvRS			ALvUS			ALvRUS		
Λ		PreSLD	PostSLD		PreSLD	PostSLD		PreSLD	PostSLD		PreSLD	PostSLD	
Λ_1	Low	26,230	26,288‡	+0.22%	25,796	12,125	-112.74%	21,395	13,106	-63.24%	10,834	14,541	+25.49%
	Med-Low	40,001	40,056‡	+0.14%	44,419	23,677	-87.60%	35,639	26,164	-36.22%	20,245	25,562	+20.80%
	Med-High	38,936	38,943‡	+0.02%	32,438	25,167	-28.89%	28,872	24,368	-18.48%	16,885	29,179	+42.13%
	High	49,190	49,193‡	+0.01%	54,839	53,807‡	-1.92%	44,137	37,600	-17.38%	31,597	64,734	+51.19%
Λ_2	Low	8,955	9,002‡	+0.53%	5,262	5,668‡	+7.16%	3,607	4,289‡	+15.90%	3,149	7,062	+55.40%
	Med-Low	12,449	12,446‡	-0.03%	10,715	11,378‡	+5.82%	6,948	9,386‡	+25.97%	5,639	13,329	+57.69%
	Med-High	14,426	14,433‡	+0.05%	9,638	12,689	+24.05%	6,972	6,288	-10.88%	5,641	12,642	+55.37%
	High	16,357	16,412‡	+0.33%	15,976	26,426	+39.54%	9,355	9,234‡	-1.32%	8,634	25,414	+66.02%
Λ_3	Low	3,181	3,195‡	+0.45%	1,383	958	-44.43%	1,951	2,050‡	+4.83%	963	1,869	+48.46%
	Med-Low	4,501	4,500‡	-0.03%	2,442	1,826	-33.74%	3,087	5,097‡	+39.43%	1,674	1,840	+9.01%
	Med-High	5,603	5,602‡	-0.03%	2,459	2,084	-18.02%	3,669	9,310‡	+60.59%	1,851	6,879	+73.08%
	High	7,986	7,970‡	-0.19%	4,682	10,072‡	+53.51%	6,289	17,350‡	+63.75%	4,025	24,406	+83.50%

Table 9
Same as Table 8, with privilege classes binned by prevalence.

	Bin	PL			ALvRS			ALvUS			ALvRUS		
Λ		PreSLD	PostSLD		PreSLD	PostSLD		PreSLD	PostSLD		PreSLD	PostSLD	
Λ_1	Low	30,904	31,097‡	+0.62%	30,142	19,869	-51.704%	24,815	18,459	-34.426%	14,180	25,302	+43.956%
	Med-Low	41,944	41,837‡	-0.26%	46,985	34,855	-34.800%	37,836	31,207†	-21.240%	23,771	45,192	+47.401%
	Med-High	41,004	41,159‡	+0.38%	42,307	34,937†	-21.095%	33,776	23,078	-46.356%	21,788	39,309	+44.572%
	High	39,971	39,921‡	-0.12%	38,626	26,842	-43.901%	33,216	27,003	-23.005%	19,935	27,806	+28.305%
Λ_2	Low	11,284	11,313‡	+0.25%	8,215	9,759†	+15.81%	4,385	4,698‡	+6.67%	3,833	10,025	+61.76%
	Med-Low	14,111	14,148‡	+0.26%	12,914	17,108†	+24.52%	7,221	7,871‡	+8.26%	6,281	17,933	+64.97%
	Med-High	13,984	14,020‡	+0.26%	10,070	17,376	+42.04%	6,109	6,634‡	+7.92%	5,407	17,559	+69.21%
	High	12,957	12,970‡	+0.10%	10,303	12,921	+20.26%	8,224	8,954‡	+8.16%	6,861	13,732	+50.03%
Λ_3	Low	4,559	4,564‡	+0.11%	2,113	2,106†	-0.31%	3,030	11,085‡	+72.66%	1,642	10,235†	+83.952%
	Med-Low	5,791	5,776‡	-0.25%	3,300	5,659†	+41.68%	4,542	17,351‡	+73.82%	2,687	18,538	+85.50%
	Med-High	5,996	6,003‡	+0.12%	3,158	6,252†	+49.48%	4,362	4,219‡	-3.40%	2,578	7,388	+65.10%
	High	5,119	5,119‡	-0.00%	2,546	2,120	-20.10%	3,346	3,144‡	-6.44%	1,823	2,011	+9.35%

Table 10
Same as Table 8, but with different policies for generating the training sets.

	Bin	Rand(RS)			Rand(US)			Rand(RUS)		
Λ		PreSLD	PostSLD		PreSLD	PostSLD		PreSLD	PostSLD	
Λ_1	Low	40,902	7,773	-426%	28,551	11,200	-155%	26,616	6,875	-287%
	Med-Low	67,945	15,605	-335%	47,435	20,328	-133%	46,674	13,261	-252%
	Med-High	56,956	13,192	-332%	37,320	19,270	-94%	36,330	10,872	-234%
	High	86,633	21,005	-312%	51,832	28,479	-82%	55,571	18,235	-205%
Λ_2	Low	8,207	1,162	-606%	4,915	2,366	-107%	5,757	1,513	-280%
	Med-Low	15,548	2,696	-476%	8,843	4,395	-101%	10,713	3,137	-241%
	Med-High	14,615	2,519	-480%	8,409	4,914	-71%	9,494	3,022	-214%
	High	20,463	3,969	-415%	11,420	6,887	-65%	13,312	4,588	-190%
Λ_3	Low	2,218	779	-185%	2,217	1,306	-70%	1,515	721	-110%
	Med-Low	3,939	1,599	-146%	3,545	2,404	-47%	2,718	1,407	-93%
	Med-High	3,958	1,697	-133%	3,998	3,011	-33%	2,741	1,553	-76%
	High	6,291	2,870	-119%	6,587	4,704	-40%	4,881	2,663	-83%

From these equations we notice that

$$\lim_{\hat{\Pr}_{\mathcal{U}}^{(s)}(y)\to 0} \Pr^{(s)} = 0 \tag{8}$$

That is, when the average of $\Pr(y=1|\mathbf{x})$ is close to 0, then we iteratively drag the distribution towards 0. Indeed, this would be the maximisation of the expectation we can make given the data: i.e., there is likely no positive item remaining (which is also not too far from the truth for the ALvRS and ALvRUS policies, as we can see from the test class prevalence value (Te prev.) in the plots). Yet, this also results in pushing an already skewed posteriors distribution even more toward the extreme where most of the probability mass already lies, producing a distribution of probabilities composed almost entirely of zero values.

SLD does not skew the posteriors distribution in case of more balanced classes, e.g., for the C17 and CCAT classes, at least when the $\mathcal{L}$ is still small, e.g., $|\mathcal{L}| = 2000$ (see Fig. 3). In these cases SLD does not

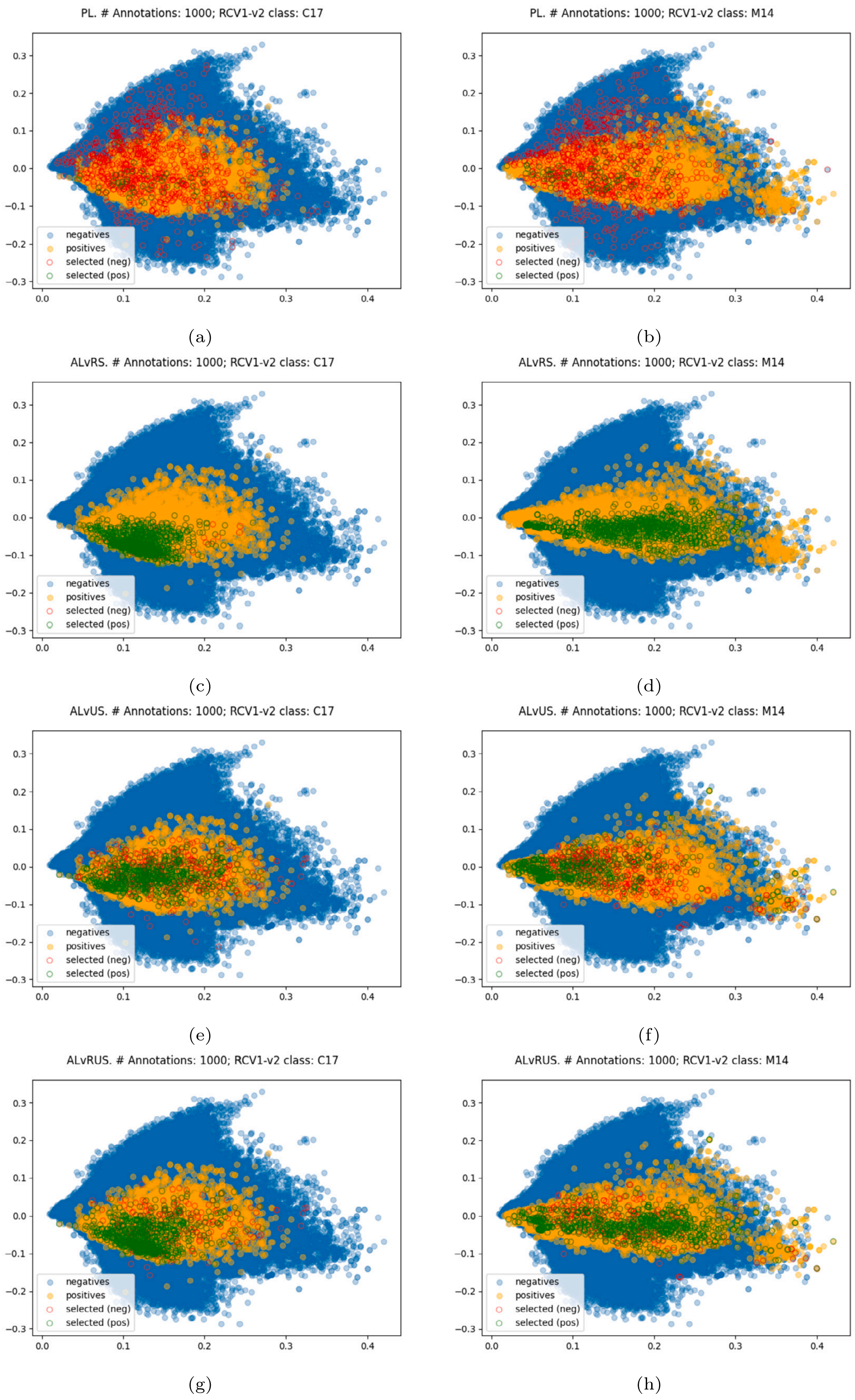

Fig. 1. First 1000 items selected by the different active learning and passive learning policies (indicated in the captions above the individual plots) for the C17 and M14 RCV1-v2 classes.

Table 11
Same as Table 9, but with different policies for generating the training sets.

	Bin	Rand(RS)			Rand(US)			Rand(RUS)		
Λ		PreSLD	PostSLD		PreSLD	PostSLD		PreSLD	PostSLD	
Λ_1	Low	46,015	8,946	-414%	32,852	12,249	-168%	30,473	7,605	-301%
	Med-Low	74,367	15,846	-369%	47,843	21,558	-122%	49,346	13,611	-262%
	Med-High	70,036	14,447	-385%	42,347	19,984	-112%	44,094	12,380	-256%
	High	62,800	16,899	-272%	41,751	23,436	-78%	41,359	14,431	-187%
Λ_2	Low	8,903	1,153	-672%	5,820	2,237	-160%	6,440	1,455	-343%
	Med-Low	16,300	2,445	-566%	9,509	4,443	-114%	11,479	2,941	-290%
	Med-High	14,419	2,298	-527%	7,624	4,337	-76%	9,060	2,759	-228%
	High	17,538	3,760	-366%	9,735	6,484	-50%	11,300	4,350	-160%
Λ_3	Low	2,889	1,153	-150%	3,334	1,910	-75%	2,149	1,077	-100%
	Med-Low	4,569	2,025	-126%	4,761	3,295	-44%	3,466	1,862	-86%
	Med-High	4,439	1,740	-155%	4,624	3,044	-52%	3,310	1,634	-102%
	High	4,385	1,917	-129%	3,827	3,070	-25%	2,963	1,703	-74%

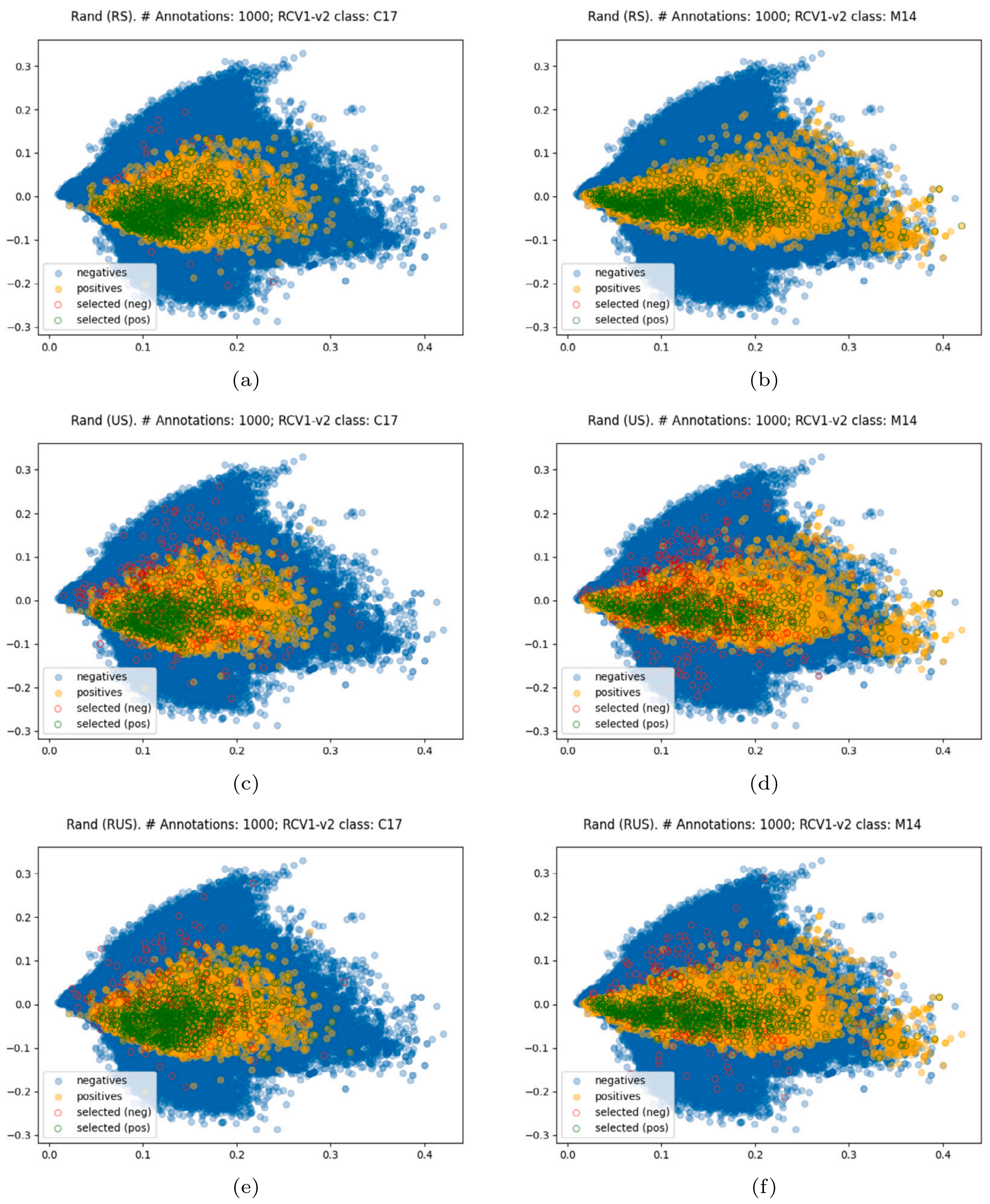

Fig. 2. Same as Fig. 1, but with the Rand policies in place of the original passive and active learning policies.

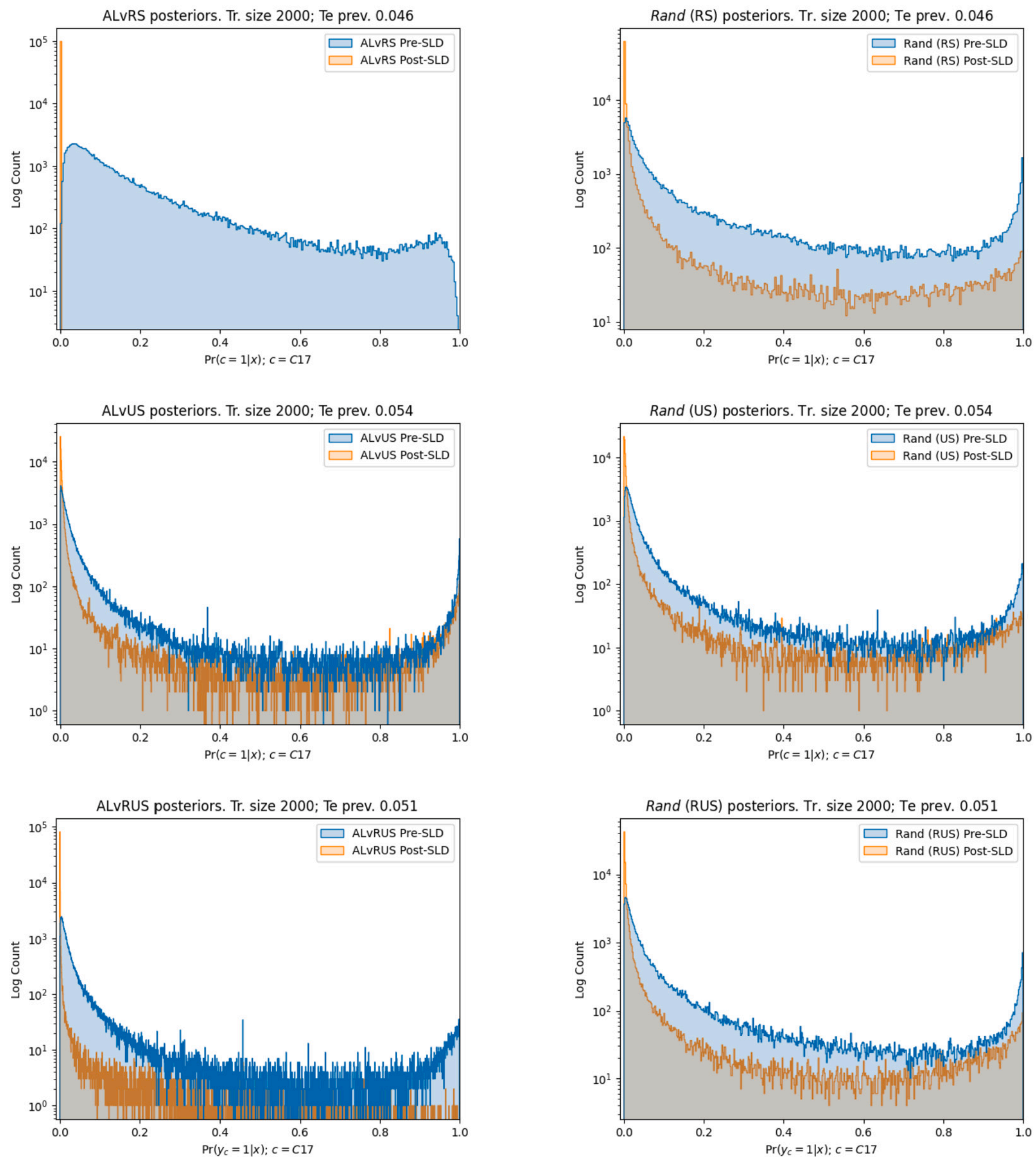

Fig. 3. Distribution of the posteriors of the unlabelled documents generated by classifiers trained with various training document selection policies (indicated in the captions above each subfigure), using a set of $|\mathcal{L}| = 2{,}000$ training documents.

shift the distribution towards its extremes, but it is also seemingly not doing much (indeed, for the ALvUS the two distributions are almost indistinguishable from each other). Notice that CCAT is one of the most populated RCV1-v2 classes. This means that, even when we draw many positives from $\mathcal{P}$, we are still not going to generate a too extreme PPS in the first iterations. Indeed, most of the positive documents would still be in the unlabelled set $\mathcal{U}$. However, this also brings SLD outside of its main scope of application, i.e., correcting high PPS: the algorithm will thus bring no significant benefit.

So, if the Rand and AL strategies are working with the same training/test class prevalence values, why are the classifier output probabilities so much more skewed in the latter case and not in the former? In order to understand this, we need to consider the sampling bias (which, again, is the main difference between the Rand and the AL policies): as a matter of fact, due to how the policies work (especially for ALvRS) we will tend to annotate many positive but similar items (see Fig. 1) and the classifier will be trained on these positive examples only. This in turn will result in the classifier being particularly good at classifying those type of positives, as well as being particularly sure of the negative label of the other documents; since the Rand policies do not suffer from sampling bias, this does not happen with these pseudo-oracle policies. This is indeed what we see in Fig. 4.

If we consider the AL strategy used and the overall prevalence value of the class in $\mathcal{P}$ we can then predict when SLD will perform a correct rescaling of the posterior probabilities or not[15]:

1. If the prevalence in $\mathcal{P}$ is low, and we use a strategy based on relevance sampling, we will remain with a very low number of positive items in the unlabelled set. Plus, since our classifier is suffering from sampling bias, its predictions will be particularly skewed towards the negative class.
2. If the prevalence in $\mathcal{P}$ is low, and we use a strategy based on uncertainty sampling, we expect to have less skewed distributed posteriors up until a certain size of annotated documents. Eventually, though, the number of positives will shrink and, with our classifier still suffering from sampling bias, we will end up in the previous scenario.

[15] Of course this is not possible in real scenarios, where we do not know $\Pr(y)$.

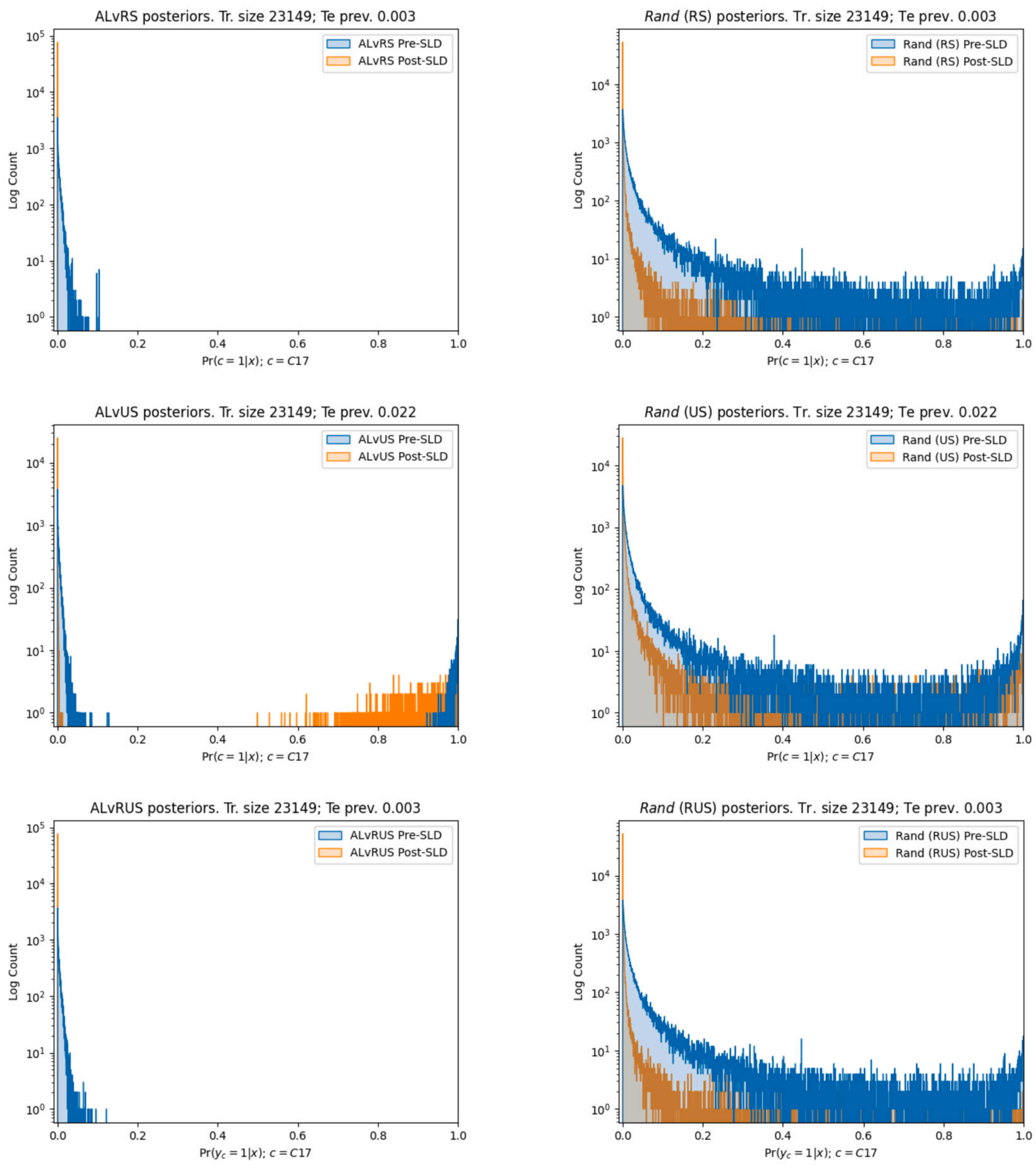

Fig. 4. As Fig. 3, but with 23,149 training documents instead of 2,000.

3. Finally, if the prevalence in $\mathcal{P}$ is fairly high and the number of annotations relatively low, then we expect the posteriors from the ALvRS/ALvRUS policy to be not very skewed.

SLD will eventually fail in all three of these scenarios, either because we have no PPS between the labelled and unlabelled sets or because sooner or later our classifiers will suffer from a strong sampling bias, which paired to the shrinking number of positives, will bring to the "posterior extremisation" phenomenon that we witness in the plots.

We then conclude this analysis arguing that SLD cannot be used "as is" in contexts where the training and test sets are resulting from AL strategies such as ALvRS, ALvRUS and ALvUS. We propose to further investigate this issue in future works, to explore possible solutions that might enable the usage of the SLD algorithm.

6. Related work

Active learning plays a central role in technology-assisted review, and researchers have long agreed on the superiority (at least when one-phase TAR systems are concerned) of AL strategies over annotating random samples from the data pool (Cormack & Mojdeh, 2009, Cormack & Grossman, 2014). The best-known and most frequently used strategy in one-phase TAR is Cormack and Grossman's Continuous Active Learning (CAL) (Cormack & Grossman, 2015b), a technique based on relevance feedback (Rocchio, 1971) and ALvRS. However, consistently with typical e-discovery scenarios, in CAL one often assumes to have no training (i.e., seed) documents. The AL process can thus be kick-started using a single positive document (which can also simply consist of the text of a query that specifies the topic of interest) and a few "negative" documents, which are actually random documents sampled from the pool to which the system automatically assigns the "non-relevant" label. The batch size can be variable, e.g., monotonically increasing as a function of the number of iterations. In the above case, the process may be stopped using a stopping heuristics that attempts to determine when a predefined level of recall has been reached. For this, a number of different heuristics have been proposed, such as the "Knee" and "Budget Knee" methods (Cormack & Grossman, 2016, Satopaa et al., 2011).

The research on stopping heuristics has intersected the research on sampling techniques for active learning. For instance, in Callaghan and Müller-Hansen's (Callaghan & Müller-Hansen, 2020) (CMH) method an initial application of the ALvRS strategy (in which the goal is to keep reviewing documents until a target recall has been achieved with a cer-

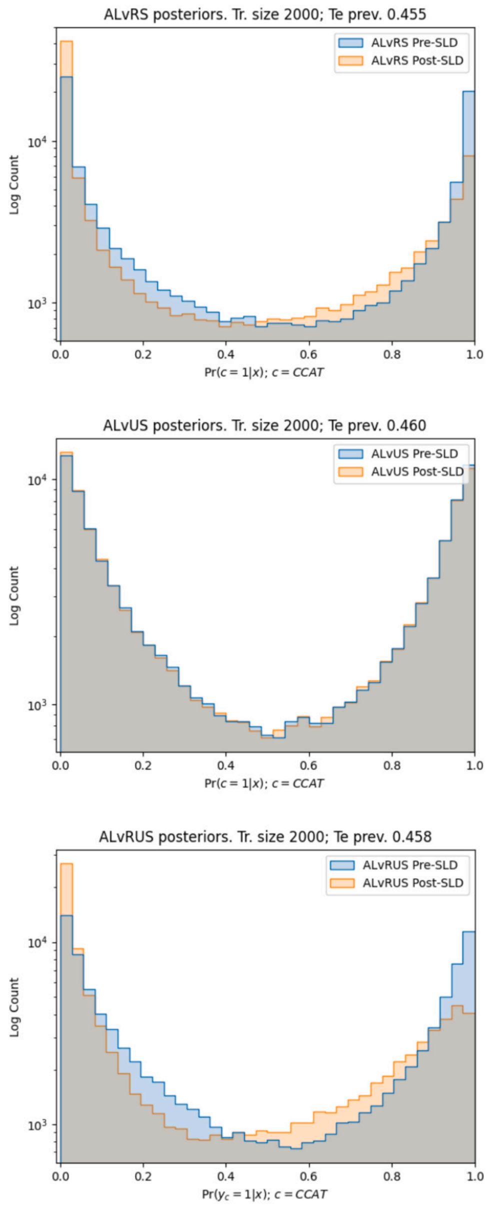

Fig. 5. ALvRS, ALvUS, and ALvRUS posteriors for the positive class, before and after the application of SLD. $|\mathcal{L}| = 2000$.

tain confidence level) is followed by a phase of random sampling, which is only stopped when the estimated recall matches the target recall with a higher confidence level.

Another recently proposed sampling (and stopping) technique is Li and Kanoulas' "autostop" framework (Li & Kanoulas, 2020); this algorithm was proposed for applying TAR to supporting the production of systematic reviews in empirical medicine, but it is suitable for applications of TAR to e-discovery as well (see also Lease et al., 2016). The review process starts, as usual, with a seed set $\mathcal{S}$, consisting of the textual description of the topic of the systematic review. At each iteration, $\mathcal{S}$ is augmented with k documents randomly sampled from the pool $\mathcal{P}$. A classifier is trained on this initial seed set, and a sampling distribution (in particular, the AP-prior distribution, see Li & Kanoulas, 2020, §3.2) is built based on the classifier's ranking, which is then used to sample a batch of documents. This process continues until stopped, according to a more or less optimistic (i.e., higher or lower confidence) strategy; we refer the reader to Li and Kanoulas (2020) for a more detailed and formal explanation.

7. Conclusions

In this work we have explored and analysed different strategies for improving the performance of the MINECORE risk minimisation framework for technology-assisted review in e-discovery (Oard et al., 2018). Specifically, we have concentrated on strategies for improving the posterior probabilities that Step 1 of the MINECORE workflow provides as input to Step 2 of the same workflow, an improvement that we measure in terms of reduction in the overall cost of the review process that MINECORE brings about. We have formulated two research questions (RQ1 and RQ2), that correspond to two possible strategies for improving these probabilities.

RQ1 poses the problem of which policy is the best for training the two classifiers that return these posterior probabilities; the policies we consider are passive learning (PL), active learning via relevance sampling (ALvRS), active learning via uncertainty sampling (ALvUS), and a combination of the two latter policies that we call active learning via relevance and uncertainty sampling (ALvRUS). The results of our experiments show that ALvRUS is unquestionably the best such policy, thus indicating that reaching a balance between "exploration" (the US component) and "exploitation" (the RS component) proves a key step in generating better training sets for MINECORE. Passive learning proves instead the worst such policy, which confirms the analogous results obtained for one-phase TAR systems (see Cormack & Grossman, 2014).

In RQ2 we instead pose the problem whether an application of the well-known SLD algorithm (Saerens et al., 2002), whose goal is to improve the quality of the posterior probabilities in contexts affected by prior probability shift (PPS), could indeed prove beneficial for MINECORE. Here, the results are less uniform, and show that the application of SLD often decreases (instead of increasing) the quality of the posterior probabilities, especially when some active learning policy has been used to train the classifiers; unfortunately, SLD always brings about a deterioration in the quality of these probabilities when ALvRUS (that had proved the "winning" policy for RQ1) has been used to train the classifiers. Additional experiments that we have run unequivocally show that the reason why SLD tends to perform badly when the classifiers have been trained via active learning, is that active learning generates not only prior probability shift (which SLD has been designed for) but also sampling bias (for which SLD is not equipped). This raises the question whether using other active learning strategies that attempt to maximise diversity / representativeness of the training data (thereby doing away with sampling bias) could allow SLD to be used profitably; this proves a difficult path to follow, though, since active learning techniques that maximise diversity tend to be too expensive, from a computational point of view, for the typical problem sizes encountered in TAR.

However, when we take RQ1 and RQ2 altogether, the answer returned by our experiments is unequivocal: the best course of action consists of (i) using ALvRUS for training the classifiers, and (ii) *not* using SLD in the attempt to further improve the posterior probabilities.

In future work we would like to investigate (for RQ1) the use of active learning techniques that are both computationally efficient and diversity-preserving, as well as (for RQ2) variants of the SLD algorithm that are also robust to the presence of sampling bias.

CRediT authorship contribution statement

Alessio Molinari: Formal analysis; Investigation; Methodology; Resources; Software; Visualization; Roles/Writing - original draft; Writing – review & editing. **Andrea Esuli:** Formal analysis; Funding acquisition; Investigation; Methodology; Project administration; Supervision; Validation. **Fabrizio Sebastiani:** Formal analysis; Funding acquisition; Investigation; Methodology; Project administration; Resources; Supervision; Validation; Roles/Writing – original draft; Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data and code are publicly available.

Acknowledgements

The work of the authors has been supported by the SoBigData++ project, funded by the European Commission (Grant 871042) under the H2020 Programme INFRAIA-2019-1, by the AI4Media project, funded by the European Commission (Grant 951911) under the H2020 Programme ICT-48-2020, and by the SoBigData.it project, funded by the Italian Ministry of University and Research under the National Recovery and Resilience Plan (PNRR). The authors' opinions do not necessarily reflect those of the funding agencies.

Appendix A. An overview of the SLD algorithm

```
Input  : Class priors $\Pr_{\mathcal{L}}(y)$ on $\mathcal{L}$, for all $y \in \mathcal{Y}$;
         Posteriors $\Pr(y|\mathbf{x})$, for all $y \in \mathcal{Y}$ and for all $\mathbf{x} \in \mathcal{U}$;
Output: Estimates $\hat{\Pr}_{\mathcal{U}}(y)$ of class prevalence values on $\mathcal{U}$, for all $y \in \mathcal{Y}$;
         Updated posteriors $\Pr(y|\mathbf{x})$, for all $y \in \mathcal{Y}$ and for all $\mathbf{x} \in \mathcal{U}$;

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

Algorithm 1: The SLD algorithm (Saerens et al., 2002).

Algorithm 1 reports the pseudocode for SLD. We assume a training set $\mathcal{L}$ of labelled examples and a set $\mathcal{U} = \{(\mathbf{x}_1, t(\mathbf{x}_1)), \ldots, (\mathbf{x}_{|\mathcal{U}|}, t(\mathbf{x}_{|\mathcal{U}|}))\}$ of unlabelled examples, i.e., examples whose true labels $t(\mathbf{x}) \in \mathcal{Y} = \{y_1, \ldots, y_n\}$ are unknown to the system. Essentially, SLD iteratively updates (Line 11) the estimates of the class priors by using the posteriors computed in the previous iteration, and updates (Line 13) the posteriors by using the estimates of the class priors computed in the current 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 are such that Equation (6) holds. As remarked in Section 3.2, Equation (6) is a necessary (albeit not sufficient – see Esuli et al., 2021, Appendix A) condition for the posteriors $\Pr(y|\mathbf{x})$ of the documents $\mathbf{x} \in \mathcal{U}$ to be calibrated.

The algorithm iterates until convergence (Line 8), i.e., until the class priors become stable and Equation (6) 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.,[16]

$$\mathrm{AE}(\hat{\pi}^{(s-1)}_{\mathcal{U}}, \hat{\pi}^{(s)}_{\mathcal{U}}) = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} |\hat{\Pr}^{(s)}_{\mathcal{U}}(y) - \hat{\Pr}^{(s-1)}_{\mathcal{U}}(y)| \tag{A.1}$$

where $\pi^{(s)}_{\mathcal{U}}$ represents the distribution of the classes on $\mathcal{U}$ at iteration s. Alexandari et al. (2020) have recently shown that the maximum likelihood function optimised by SLD is concave, and that SLD thus converges to a global maximum.

While SLD is a natively multiclass algorithm, in this paper we restrict our analysis to the binary case, with y equal to y_r or to y_p.

References

Alexandari, A., Kundaje, A., & Shrikumar, A. (2020). Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. In *Proceedings of the 37th international conference on machine learning (ICML 2020), virtual event* (pp. 222–232).

Callaghan, M. W., & Müller-Hansen, F. (2020). Statistical stopping criteria for automated screening in systematic reviews. *Systematic Reviews*, *9*(1), 1–14.

Chhatwal, R., Keeling, R., Gronvall, P., Huber-Fliflet, N., Zhang, J., & Zhao, H. (2020). CNN application in detection of privileged documents in legal document review. In *Proceedings of the 9th IEEE international conference on big data* (pp. 1485–1492).

Cormack, G. V., & Grossman, M. R. (2014). Evaluation of machine-learning protocols for technology-assisted review in electronic discovery. In *Proceedings of the 37th ACM conference on research and development in information retrieval (SIGIR 2014)* (pp. 153–162).

Cormack, G. V., & Grossman, M. R. (2015a). Multi-faceted recall of continuous active learning for technology-assisted review. In *Proceedings of the 38th ACM conference on research and development in information retrieval (SIGIR 2015)* (pp. 763–766).

Cormack, G. V., & Grossman, M. R. (2015b). Autonomy and reliability of continuous active learning for technology-assisted review. arXiv:1504.06868 [cs.IR].

Cormack, G. V., & Grossman, M. R. (2016). Engineering quality and reliability in technology-assisted review. In *Proceedings of the 39th ACM conference on research and development in information retrieval (SIGIR 2016)* (pp. 75–84).

Cormack, G. V., & Mojdeh, M. (2009). Machine learning for information retrieval: TREC 2009 web, relevance feedback and legal tracks. In *Proceedings of the 18th text retrieval conference (TREC 2009)*.

Dasgupta, S., & Hsu, D. (2008). Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on machine learning (ICML 2008)* (pp. 208–215).

Degnan, D. (2011). Accounting for the costs of electronic discovery. *Minnesota Journal of Law, Science and Technology*, *12*(1), 151–190.

Esuli, A., Moreo, A., & Sebastiani, F. (2019). Building automated survey coders via interactive machine learning. *International Journal of Market Research*, *61*(4), 408–429. https://doi.org/10.1177/1470785318824244.

Esuli, A., Molinari, A., & Sebastiani, F. (2021). A critical reassessment of the Saerens-Latinne-Decaestecker algorithm for posterior probability adjustment. *ACM Transactions on Information Systems*, *39*(2), Article 19. https://doi.org/10.1145/3433164.

Flach, P. A. (2017). Classifier calibration. In C. Sammut, & G. I. Webb (Eds.), *Encyclopedia of machine learning* (2nd edition) (pp. 212–219). Heidelberg, DE: Springer.

Grossman, M. R., & Cormack, G. V. (2011). Technology-assisted review in e-discovery can be more effective and more efficient than exhaustive manual review. *Richmond Journal of Law and Technology*, *17*(3), Article 5.

Huang, S., Jin, R., & Zhou, Z. (2014). Active learning by querying informative and representative examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(10), 1936–1949. https://doi.org/10.1109/TPAMI.2014.2307881.

Kanoulas, E., Li, D., Azzopardi, L., & Spijker, R. (2019). CLEF 2019 technology assisted reviews in empirical medicine overview. In *Working notes of the conference and labs of the evaluation forum (CLEF 2019)*.

Krishnan, R., Sinha, A., Ahuja, N., Subedar, M., Tickoo, O., & Iyer, R. (2021). Mitigating sampling bias and improving robustness in active learning. Preprint. arXiv: 2109.06321.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*, *25*(2–3), 259–284.

[16] Consistently with most mathematical literature, we use the caret symbol (ˆ) to indicate estimation.

Lease, M., Cormack, G. V., Nguyen, A. T., Trikalinos, T. A., & Wallace, B. C. (2016). Systematic review is e-discovery in doctor's clothing. In *Proceedings of the SIGIR 2016 medical information retrieval workshop (MedIR 2016)*.

Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th ACM international conference on research and development in information retrieval (SIGIR 1994)* (pp. 3–12).

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, *5*, 361–397.

Li, D., & Kanoulas, E. (2020). When to stop reviewing in technology-assisted reviews: Sampling from an adaptive distribution to estimate residual relevant documents. *ACM Transactions on Information Systems*, *38*(4), 41:1–41:36. https://doi.org/10.1145/3411755.

Moreno-Torres, J. G., Raeder, T., Alaíz-Rodríguez, R., Chawla, N. V., & Herrera, F. (2012). A unifying view on dataset shift in classification. *Pattern Recognition*, *45*(1), 521–530. https://doi.org/10.1016/j.patcog.2011.06.019.

Oard, D. W., & Webber, W. (2013). Information retrieval for e-discovery. *Foundations and Trends in Information Retrieval*, *7*(2/3), 99–237. https://doi.org/10.1561/1500000025.

Oard, D. W., Sebastiani, F., & Vinjumur, J. K. (2018). Jointly minimizing the expected costs of review for responsiveness and privilege in e-discovery. *ACM Transactions on Information Systems*, *37*(1), 11:1–11:35. https://doi.org/10.1145/3268928.

O'Mara-Eves, A., Thomas, J., McNaught, J., Miwa, M., & Ananiadou, S. (2015). Using text mining for study identification in systematic reviews: A systematic review of current approaches. *Systematic Reviews*, *4*(5), 1–22.

Patel, S., Sihmar, S., & Jatain, A. (2015). A study of hierarchical clustering algorithms. In *2015 2nd international conference on computing for sustainable global development (INDIACom)* (pp. 537–541).

Platt, J. C. (2000). 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* (pp. 61–74). Cambridge, MA: The MIT Press.

Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (Eds.). (2009). *Dataset shift in machine learning*. Cambridge, US: The MIT Press.

Rocchio, J. J. (1971). Relevance feedback in information retrieval. In G. Salton (Ed.), *The SMART retrieval system: Experiments in automatic document processing* (pp. 313–323). Englewood Cliffs, US: Prentice-Hall.

Roitblat, H. L., Kershaw, A., & Oot, P. (2010). Document categorization in legal electronic discovery: Computer classification vs. manual review. *Journal of the American Society for Information Science and Technologies*, *61*(1), 70–80. https://doi.org/10.1002/asi.21233.

Saerens, M., Latinne, P., & Decaestecker, C. (2002). Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, *14*(1), 21–41. https://doi.org/10.1162/089976602753284446.

Satopaa, V., Albrecht, J., Irwin, D., & Raghavan, B. (2011). Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *Proceedings of the 31st IEEE international conference on distributed computing systems (ICDCS 2011) workshops* (pp. 166–171).

Settles, B. (2012). *Active learning*. San Rafael, US: Morgan & Claypool Publishers.

Storkey, A. (2009). When training and test sets are different: Characterizing learning transfer. In J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, & N. D. Lawrence (Eds.), *Dataset shift in machine learning* (pp. 3–28). Cambridge, US: The MIT Press.

Vinjumur, J. K. (2018). Predictive coding techniques with manual review to identify privileged documents in e-discovery. Ph.D. thesis, College Park, US: University of Maryland.

Yang, E., Lewis, D. D., & Frieder, O. (2021a). TAR on social media: A framework for online content moderation. In *Proceedings of the 2nd international conference on design of experimental search & information REtrieval systems (DESIRES 2021)* (pp. 147–155).

Yang, E., Lewis, D. D., & Frieder, O. (2021b). Heuristic stopping rules for technology-assisted review. In *Proceedings of the 21st ACM symposium on document engineering (DocEng 2021)* (pp. 31:1–31:10).

Yang, E., Lewis, D. D., & Frieder, O. (2021c). On minimizing cost in legal document review workflows. In *Proceedings of the 21st ACM symposium on document engineering (DocEng 2021)* (pp. 1–10).

Zhao, H., Ye, S., & Yang, J. (2021). An empirical study on transfer learning for privilege review. In *Proceedings of the 10th IEEE international conference on big data* (pp. 2729–2733).