AnchorAL: Computationally Efficient Active Learning for Large and Imbalanced Datasets

Pietro Lesci **Andreas Vlachos**
Department of Computer Science and Technology
University of Cambridge
{pl487, av308}@cam.ac.uk

Abstract

Active learning for imbalanced classification tasks is challenging as the minority classes naturally occur rarely. Gathering a large pool of unlabelled data is thus essential to capture minority instances. Standard pool-based active learning is computationally expensive on large pools and often reaches low accuracy by overfitting the initial decision boundary, thus failing to explore the input space and find minority instances. To address these issues we propose AnchorAL. At each iteration, AnchorAL chooses class-specific instances from the labelled set, or *anchors*, and retrieves the most similar unlabelled instances from the pool. This resulting *subpool* is then used for active learning. Using a small, fixed-sized subpool AnchorAL allows scaling any active learning strategy to large pools. By dynamically selecting different anchors at each iteration it promotes class balance and prevents overfitting the initial decision boundary, thus promoting the discovery of new clusters of minority instances. Experiments across different classification tasks, active learning strategies, and model architectures AnchorAL is *(i)* faster, often reducing runtime from hours to minutes, *(ii)* trains more performant models, *(iii)* and returns more balanced datasets than competing methods.

github.com/pietrolesci/anchoral

1 Introduction

The abundance of web-scale textual data[1] has contributed to the success of generalist language models pretrained as multi-purpose foundation models and fine-tuned to solve downstream natural language processing tasks (Bommasani et al., 2022). The data used during fine-tuning critically affects their downstream abilities (Lee et al., 2022; Gururangan et al., 2022), especially in real-world applications where performance on rare *concepts*, or minority classes, is critical (He and Garcia, 2009).

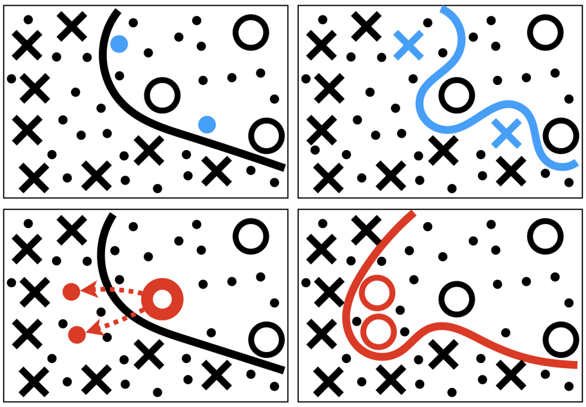

Figure 1: AnchorAL (intuition). Binary classification task where ○, ×, and ● are labelled minority, labelled majority, and unlabelled instances. The black (left) and coloured (right) lines denote the initial and final decision boundary. Typical AL (top) selects instances near the current boundary. AnchorAL (bottom) anchors the selection to labelled instance(s) (bold red ○) and discovers a new minority cluster.

Data collection and annotation for imbalanced classification tasks is challenging as the minority class(es) naturally occur(s) rarely. Gathering a large pool of unlabelled data is often essential to capture minority instances making manual annotation prohibitively expensive. In principle, considering only a random subset of the pool is potentially amenable to manual curation but can be suboptimal when the imbalance is large as instances of the rare class might never be found. Therefore, generally, the learning challenges caused by the imbalance and the computational challenges stemming from considering a large pool are intrinsically intertwined and need to be jointly addressed. Active Learning (AL) provides an automatic mechanism to prioritise instances by allowing a model to select those to label, resulting in a higher label efficiency and lower annotation costs (Hanneke, 2007, 2011; Balcan et al., 2009, 2010). Also, it can partially mitigate mild imbalance (Ertekin et al., 2007).

[1] E.g. Common Crawl corpus is in the order of petabytes.

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 8438–8457
June 16-21, 2024 ©2024 Association for Computational Linguistics

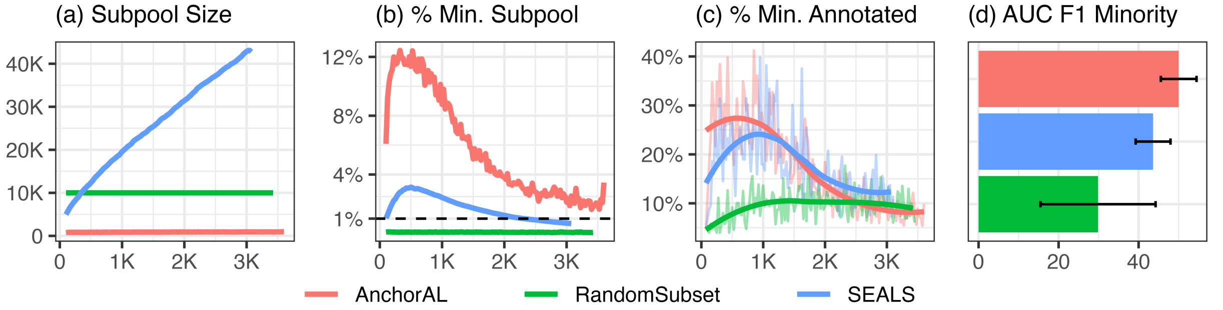

Figure 2: AnchorAL (this paper) vs RandSub (Ertekin et al., 2007) and SEALS (Coleman et al., 2022) on an imbalanced ($<1\%$) binary classification task (Amazon-Agri) for the ALBERT-base model and the Entropy strategy. AnchorAL keeps the subpool size fixed and small across iterations (x-axis) (a) which reduces computational and annotation costs (i.e., annotators' waiting time). The subpool is more balanced (b) and allows the AL strategy to discover minority instances earlier (c) and reach better performance (d).

Standard pool-based active learning (AL), however, struggles with large and imbalanced pools. First, AL can be too computationally expensive due to its iterative nature. Specifically, the inference costs of repeatedly evaluating a model on every unlabelled instance in each iteration can be prohibitive, especially given the size of modern language models (Tsvigun et al., 2022); furthermore, the annotation costs (i.e., the man-hours per annotation) can drastically increase due to the annotators' waiting time between iterations. Second, AL can be as good as random selection due to the imbalance: diversity-based strategies can be ineffective when minority and majority instances are not easily separable in high-dimensional spaces (Thudumu et al., 2020), while uncertainty-based strategies can fail to explore the input space and discover minority clusters because they tend to keep refining the initial decision boundary (Tomanek et al., 2009); hybrid strategies suffer from similar limitations. One line of prior work has proposed to subset the pool in each iteration before running an AL strategy to speed up instance selection (Ertekin et al., 2007; Coleman et al., 2022). However, existing pool filtering methods either do not address class imbalance or are computationally inefficient. Therefore, it is still an open question how to scale AL on large datasets while addressing the imbalance.

In this paper, we propose a new method, AnchorAL, designed to jointly address the learning and computational challenges of applying AL to large and imbalanced datasets. AnchorAL works by filtering the pool before running any AL strategy, thus reducing the inference costs and the annotators' waiting time in each iteration. Crucially, the filtering process is designed to promote the exploration of the input space and the discovery of minority instances while keeping instance selection time constant, regardless of the original pool size. Specifically, at each iteration, AnchorAL chooses class-specific instances from the labelled set, which we term *anchors*. Then, each unlabelled instance is scored based on its average distance from the anchors and the most similar are used to form the subset of the pool, which we term *subpool*. While any similarity measure works, we use the semantic representation capabilities of language models (e.g., Devlin et al., 2019) and measure similarity based on cosine distance between instance representations. Using a small, fixed-sized subpool AnchorAL allows scaling any active learning strategy to large pools while keeping the annotators' waiting time constant across iterations independently of the size of the original pool. Optionally, the maximum size of the subpool can be set by the user, thus making AnchorAL suitable for different computational budgets. Moreover, by dynamically selecting different anchors at each iteration it prevents overfitting the initial decision boundary, thus promoting the discovery of new clusters of minority instances; the class-specific anchors promote class balance even for AL strategies that do not explicitly account for imbalance (Fig. 2).

We test the effectiveness of AnchorAL across 4 text classification tasks, both binary and multiclass, 3 AL strategies (i.e., uncertainty, diversity, and hybrid), and 6 models with different sizes and architectures (i.e., encoder, decoder, and encoder-decoder). Experiments show that AnchorAL is the fastest method, reducing the total selection time from hours to minutes; (often) the best-performing, reaching higher performance in less time and with fewer annotations; and discovers the most minority instances resulting in more balanced labelled sets.

2 Related Work

In this section, we provide the background on imbalanced learning and present the challenges of applying AL to imbalanced datasets. Then, we discuss approaches to address these challenges and scale AL to large pools.

2.1 Learning from Imbalanced Datasets

The research on imbalanced learning (Johnson and Khoshgoftaar, 2019; Henning et al., 2023) can be broadly divided into two approaches. Data-level approaches directly balance the training data distributions by generating synthetic samples (Mullick et al., 2019) or resampling the available data (Chawla et al., 2002; Van Hulse et al., 2007). Reweighting approaches assign a different weight to each instance to adjust its contribution to the training loss and thus its importance in determining the parameter updates based on the stochastic gradients (Dong et al., 2017; Wang et al., 2017; Lin et al., 2017; Cui et al., 2019; Park et al., 2021).

Although AnchorAL can be associated with data-level approaches for its re-balancing effect, it differs from those in two crucial aspects. First, the objective of AnchorAL is to choose *which* data to learn from rather than optimise *how* to learn from data. Second, AnchorAL targets a setting in which the labels are not known ex-ante, that is AL. Therefore, it cannot easily rely on data augmentation, generation, or over/under-sampling. Instead, it uses the semantic representation capabilities of language models and smartly selects anchors to use as queries to retrieve a more balanced subpool. In Table 3 we show that which and how many anchors are chosen strongly affects performance.

2.2 Class Imbalance in Active Learning

Standard pool-based AL, by design, tends to explore the regions nearest to the current decision boundary to *refine* the learned decision boundary until it eventually approaches the optimal one (Osugi et al., 2005). However, the initial approximation can be distorted and converge to a suboptimal decision boundary when the data is imbalanced. Specifically, the class distribution derived from the initial imbalanced labelled set results in a rough initial decision boundary that makes the model overconfident about predicting the majority class everywhere as the low-certainty regions are just around the minority instances initially seen (Park et al., 2021). As a result, the AL strategy fails to explore the input space (Fig. 1) and keeps refining the known decision boundary (Baram et al., 2004; Dligach and Palmer, 2011), a phenomenon that we term *path dependence*. Path dependence is exacerbated when minority instances sparsely occupy the input space instead of forming tight clusters. If the initial set contains minority instances that are only from a specific cluster the AL strategy can even exhaust its budget before discovering new clusters (Attenberg and Provost, 2010; Attenberg and Ertekin, 2013). This issue has long been studied in AL and historically known as *missed cluster effect* (Schütze et al., 2006) or *hasty generalisation* (Wallace et al., 2010).

Addressing path dependence and the missed cluster effect is especially relevant in the context of language models fine-tuning because recent large pretrained models have the potential to memorise their entire training data (Kim et al., 2023) and tend to overfit to the overlapping regions between classes (Arpit et al., 2017; Zhang et al., 2021).

2.3 Actively Learning Imbalanced Datasets

To address the class imbalance issue in AL, prior work proposes to change the interaction protocol by allowing annotators to directly search for instances to label (Attenberg and Provost, 2010; Balcan and Hanneke, 2012, *inter alia*). Similarly, the line of work in Active Search proposes to directly optimise for the recall of minority instances rather than model performance, as in typical AL (Garnett et al., 2012; Jiang et al., 2018, 2019, *inter alia*). These approaches propose interaction protocols or objectives that differ from AL and are thus out of the scope of our paper. Moreover, although search has been shown to theoretically improve label efficiency (Beygelzimer et al., 2016), in practice it is significantly more expensive since the annotation process is more challenging and thus requires more time. Also, search can be suboptimal beyond the initial data collection phase (Levonian et al., 2022), crucially depends on the annotators' ability to find useful keywords for the specific domain of interest, and might not be feasible in some cases: for example it might not be possible to search for a specific hateful *sense* of a word (Hartford et al., 2020).

Instead, our work finds its roots in Ertekin et al. (2007) that originally proposed to randomly sample a fixed-sized subset of the pool at each iteration to speed up the AL process, an approach that we refer to as RandSub. While motivated by its purely computational benefits, our intuition is that pool filtering can address the imbalance too: choosing a different subset of the pool in each iteration forces the AL strategy to explore different parts of the input space instead of focusing on refining the initially learned decision boundary. Intuitively, a more extensive exploration of the input space promotes the discovery of new minority instances. Recently, Coleman et al. (2022) showed that random subsampling can be ineffective when the imbalance is high and proposed SEALS which restricts the pool to the k-nearest neighbours of all labelled instances. However, SEALS suffers from computational inefficiencies because the size of the subpool grows throughout the AL process: when a new instance is labelled its k neighbours are added to the subpool and are not removed unless labelled. Also, when the initial labelled set is small, the resulting initial subpool is small too (when k is small) or redundant (when k is large) which makes subsequent subpools similar to each other thus limiting the exploration of the input space due to the path dependence.

AnchorAL shares the same motivation as RandSub and SEALS but proposes an alternative approach that allows keeping the subpool size fixed across iterations while still effectively promoting the exploration of the input space, thus combining the benefits of both without none of the downsides. Similarly to RandSub, AnchorAL chooses a different, fixed-sized subpool at each iteration by selecting different anchors which makes it *dynamically* adapt to the changing labelled set during AL. Instead, for SEALS the subpool size grows by k units per each new labelled instance and after a few iterations the subpool remains almost unchanged because the new instances are only a small fraction of the entire subpool. Similarly to SEALS, AnchorAL relies on similarity search to discover minority instances while RandSub can fail under extreme class imbalance. However, differently from SEALS, AnchorAL emphasises query selection (i.e., anchors) as it targets retrieval of useful instances rather than purely maximising recall of minority instances.

2.4 Active Learning on Large Pools

The computational overhead required by the iterative nature of pool-based AL when applied to large pools, especially in combination with large models, can make AL infeasible. Recent work in computationally efficient AL has proposed to use smaller models as cheap proxies (Yoo and Kweon, 2019; Coleman et al., 2020, *inter alia*), selecting large batches that are both informative and diverse to reduce the number of labelling iterations necessary to reach a target performance (Sener and Savarese, 2018; Kirsch et al., 2019; Pinsler et al., 2019, *inter alia*), or generating examples (Lin et al., 2018; Mayer and Timofte, 2020, *inter alia*). However, proxies and large-batch approaches still require evaluation over the entire pool, and generative methods struggle to match the label efficiency of traditional AL (Settles, 2012). Instead, AnchorAL directly limits the size of the pool. Besides the rebalancing effects discussed in the previous sections, this approach allows scaling up to large pools any AL strategies and models without modifications.

Advantages of AnchorAL vs Prior Work

AnchorAL promotes the exploration of the input space by dynamically selecting different anchors which then retrieve a different subpool at each iteration, thus preventing the AL strategy from overfitting to the initial labelled set. Moreover, by keeping the size of the subpool fixed and small across iterations, it can scale to large pools, independently of their original size.

3 Methodology

We consider the standard pool-based AL setting for classification tasks (Atlas et al., 1989) wherein we are given a (large) pool of unlabelled data $\mathcal{U}$ and access to an oracle $f^{\star}$ that given an instance $\boldsymbol{x}$ returns its true[2] label $y \in \{1, ..., C\}$. The goal is to induce the best possible classifier f within a fixed annotation budget B. The annotation process happens iteratively over T iterations. We assume access to a small initial labelled set $\mathcal{D}_0$ with at least one instance per class used to bootstrap an initial classifier f_0. In each iteration, we train a new version of the classifier f_t on the available labelled instances $\mathcal{D}_t$. Then, the AL strategy Φ uses the model to inform the selection of the set of instances to annotate $\mathcal{B}_t \subset \mathcal{U}_t$ such that $|\mathcal{B}_t| = \lfloor B/T \rfloor$. Finally, the selected instances are annotated by the oracle, removed from the pool, and added to the labelled set; and the cycle repeats.

3.1 AnchorAL: Anchored Pool Filtering

AnchorAL runs the AL strategy only on a subset of the pool $\widetilde{\mathcal{U}} \subset \mathcal{U}$, such that $|\widetilde{\mathcal{U}}| \leq M \ll |\mathcal{U}|$ where M is a fixed, user-defined upper bound on the sub-

[2] We do not consider noisy oracles (Settles, 2012).

Algorithm 1 AnchorAL

```
Require: AL strategy $\Phi$, anchor selection strategy $\Gamma$, per-class number of anchors $A$, per-anchor number
    of neighbours $K$, and maximum subpool size $M$.
Ensure: Unlabelled pool $\mathcal{U}_0$, initial labelled set $\mathcal{D}_0$, oracle $f^\star$, dense index $\mathcal{I}$.
 1: for $t$ to $T$ do
 2:     $f_t = f.\mathtt{train}(\mathcal{D}_t)$                                    ▷ Train model on the available data
 3:     $\mathcal{A}_t = \emptyset$
 4:     for $c$ in $\{1, ..., C\}$ do
 5:         $\mathcal{A}_t \leftarrow \Gamma\big(\{\boldsymbol{x} \mid y = c \;\; \forall\, (\boldsymbol{x}, y) \in \mathcal{D}_t\}; A\big)$     ▷ Select $A$ anchors per class
 6:     end for
 7:     $\mathcal{N}_t = \mathcal{I}.\mathtt{kNN}(\mathcal{A}_t, \mathcal{U}_t; K)$                   ▷ Retrieve $K$ neighbours from the pool
 8:     $\widetilde{\mathcal{N}}_t = \mathtt{avg}(\mathcal{N}_t)$                          ▷ Average similarity scores by instance
 9:     $\widetilde{\mathcal{U}}_t = \mathtt{argmax}(\widetilde{\mathcal{N}}_t; \bar{M})$ where $\bar{M} = \min\{|\widetilde{\mathcal{N}}|, M\}$   ▷ Keep the top-$\bar{M}$ instances
10:     $\mathcal{B}_t = \Phi(f_t, \widetilde{\mathcal{U}}_t; b)$ where $b = \lfloor B/T \rfloor$          ▷ Select $b$ instances to label from the subpool
11:     $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(\boldsymbol{x}_u, f^\star(\boldsymbol{x}_u)) \mid \boldsymbol{x}_u \in \mathcal{B}_t\}$   ▷ Label queried instances and add to training set
12:     $\mathcal{U}_{t+1} = \mathcal{U}_t \setminus \mathcal{B}_t$                           ▷ Remove labelled instances from the pool
13: end for
```

pool size. To construct the subpool $\widetilde{\mathcal{U}}$ in a way that promotes exploration and favours minority cluster discovery beyond those present in the initial set, AnchorAL anchors the filtering process to a set of labelled instances $\mathcal{A}_t \subset \mathcal{D}_t$ created selecting A instances from each class, such that $|\mathcal{A}_t| = A \times C$. These anchors are dynamically selected and differ at each iteration. Then, the unlabelled instances are scored based on their average distance from the anchors and the $\bar{M} \leq M$ most similar are used as subpool (Alg. 1). In the following paragraph we detail these two parts of the algorithm.

Intuition

AnchorAL "biases" the subpool towards smaller regions of the input space so that the resulting subpool only represents these regions and forces the AL strategy to focus on those. Since anchors are chosen according to a diversity criterion, in each iteration the resulting subpool represents a small and different region of the input space, thus promoting exploration and consequently selection of minority instances for labelling.

Anchor Selection (Lines 3 to 6). The anchor selection strategy Γ can be class-specific but using only one strategy for all classes is enough in practice. Similarly, we select the same number of anchors $A = 10$ per each class. To elicit the discovery of new minority clusters, we choose an anchor selection strategy that promotes diversity and use the k-MEANS++ initialisation scheme (Arthur and Vassilvitskii, 2007).[3] It was originally proposed to produce a good initialisation for k-MEANS clustering (Steinhaus, 1956) and works by iteratively sampling points in proportion to their squared distances from the nearest points already chosen. We run it for each class separately and apply it to the sentence representations derived as described in the next paragraph. We discuss other strategies in §6.

Similarity Scoring (Lines 7 to 8). To score unlabelled instances we need to define a similarity measure. While any similarity measure works (e.g., BM25 by Robertson and Zaragoza, 2009) we use the semantic representation capabilities of language models and measure similarity based on cosine distance between instance representations. Thus, we construct a dense index $\mathcal{I}$, that is kept fixed for the entire AL process, using a pre-trained encoder. Specifically, we use MPNet (Song et al., 2020), which is specifically trained to use cosine distances as the similarity function, to encode all available instances (i.e., $\mathcal{U}_0 \cup \mathcal{D}_0$). Generating the embeddings can be computationally expensive but it is performed once and its cost is amortised over the entire AL process. Also, often the encoder has a similar or smaller size than the model we train, thus creating the embeddings has approximately the same cost as running one iteration of standard AL.[4] Given the embeddings, we create a searchable

[3] As implemented in sklearn.cluster.kmeans_plusplus.

[4] Encoding can be sped up using efficient procedures (optimum/fast-mteb) and new encoders (mteb/leaderboard).

index $\mathcal{I}$ using the Hierarchical Navigable Small World algorithm (Malkov and Yashunin, 2018),[5] an approximate nearest-neighbour search method that can easily scale to extremely large ($>$1B) datasets with retrieval times in the milliseconds (Johnson et al., 2017; Babenko and Lempitsky, 2016).

To make the scoring mechanism efficient, instead of scoring each unlabelled instance against each anchor, we retrieve the $K = 50$ nearest neighbours of each anchor from the pool $\mathcal{N}_t \subset \mathcal{U}_t$ (Line 7) and the relative similarity scores such that $|\mathcal{N}_t| \leq A \times C \times K$, where the equality is not strict because in practice different anchors can retrieve the same unlabelled instance. When duplicates are retrieved, we average their similarity scores and get a de-duplicated set of neighbours, $\widetilde{\mathcal{N}}$ (Line 8).

Subpool Selection (Line 9). Finally, the $\bar{M} = \min\{|\widetilde{\mathcal{N}}|, M\}$ pool instances with the highest similarity score are used as the subpool $\widetilde{\mathcal{U}}$ to run the AL strategy. The hyper-parameter M controls the maximum size of the pool. In our experiments, we show that values as small as 1k are enough to achieve good performance. This step is responsible for the main speed-up provided by AnchorAL.

> Tips: Hyper-parameters Settings
>
> The number of anchors per class A and the number of neighbours retrieved per each class K should both be set to small values. Specifically, we suggest setting A to 5-20 and K to 20-50. The intuition is that as A increases the subpool tends to be similar to the labelled set, thus failing to break the path dependence. As K increases the subpool tends to be dominated by majority instances due to the imbalance, thus reducing the chance of selecting minority instances.

4 Experimental Setup

We mimic a realistic and interactive annotation setting (Grießhaber et al., 2020; Yuan et al., 2020; Maekawa et al., 2022; Schröder et al., 2022) characterised by a budget of 5k[6] annotations and label 25 instances per iteration.[7] We start with an initial set of 100 instances $\mathcal{D}_0$, containing 5 from each minority class and the rest from the majority, chosen randomly. We do not assume access to a validation set. In each iteration, we re-initialise the model and train all parameters. We limit each run (i.e., training, instance selection, and testing) to 6h to allow for thorough experimentation while complying with our computational budget.[8]

Type	Model	# Params
Encoder	BERT-base (Devlin et al., 2019)	110M
	BERT-tiny (Turc et al., 2019)	4.4M
	ALBERT-base (Lan et al., 2019)	12M
	DeBERTa-base (He et al., 2023)	86M
Decoder	GPT-2 (Radford et al., 2019)	117M
Enc-Dec	T5-base (Raffel et al., 2020)	220M

Table 1: Models overview. We use the checkpoints available on the HuggingFace Hub ((linked) and loaded using the AutoModelForSequenceClassification class.

Evaluation. We use the (macro-averaged) F1-score on the minority class(es) as our predictive accuracy metric. For completeness, we report the majority class performance, too. We evaluate the model on a held-out set after each iteration and report the area under the learning curve (AUC) computed using the trapezoidal rule.[9] Moreover, we report the total (across iterations) instance selection time as a proxy of the total annotators' waiting time, assuming uniform instance annotation difficulty (Settles, 2012). To allow for a more robust comparison, we use 2 random seeds for each major source of randomness: model initialisation, data ordering, and initial data selection resulting in 8 runs per experiment and report the median and interquartile range (Liu et al., 2022).

Baselines. We compare AnchorAL with two pool filtering methods. RandSub (Ertekin et al., 2007) samples $\bar{M}$ instances uniformly at random from the pool; we set $\bar{M} = 10\text{k}$.[10] SEALS (Coleman et al., 2022) limits the pool to the set of $K = 50$ neighbours of the currently labelled data: after every instance is selected, its K-nearest neighbours are added to the pool. Moreover, we compare against standard AL (No-Op) which considers the entire pool and random sampling (Random).

[5] As implemented in the hnswlib library (App. A.1).

[6] Throughout the paper, we use shorthands for units, that is, k (thousands), M (millions), and h (hours).

[7] Note that this setup differs from previous AL studies (Margatina et al., 2022, 2021; Citovsky et al., 2021, *inter alia*) that label a large portion of the pool per iteration (e.g., 1%) and assume a large validation set throughout the AL process.

[8] Datasets and experimental artefacts available at huggingface.co/collections/pietrolesci/anchoral.

[9] As implemented in numpy.trapz.

[10] As it would be done in a real setting, we searched over a grid of values in 5k-20k for a small budget of 200 labelled instances and selected the best (speed and AUC) $\bar{M}$.

Dataset	Pool Filtering	Overall				Budget-Matched	
		Budget	Majority	Minority	Time (↓)	Majority	Minority
Amazon-Agri	AnchorAL	3.7k	$142.4_{\pm 0.0}$	$78.1_{\pm 1.0}$	$4.4_{\pm 0.1}$	$119.8_{\pm 0.1}$	$64.9_{\pm 0.9}$
	RandSub	3.6k	$138.3_{\pm 0.1}$	$65.0_{\pm 3.2}$	$29.0_{\pm 0.2}$	$119.6_{\pm 0.1}$	$54.3_{\pm 2.0}$
	SEALS	3.2k	$123.7_{\pm 0.0}$	$65.6_{\pm 0.8}$	$85.4_{\pm 1.3}$	$119.8_{\pm 0.0}$	$63.7_{\pm 1.4}$
	No-Op	475	$14.6_{\pm 0.0}$	$2.9_{\pm 0.0}$	6h	$14.6_{\pm 0.0}$	$2.9_{\pm 0.0}$
Amazon-Multi	AnchorAL	3.6k	$109.2_{\pm 1.7}$	$77.9_{\pm 9.9}$	$7.8_{\pm 0.2}$	$92.7_{\pm 1.7}$	$65.3_{\pm 12.4}$
	RandSub	3.5k	$106.2_{\pm 0.6}$	$74.7_{\pm 14.2}$	$26.6_{\pm 0.3}$	$92.8_{\pm 0.6}$	$65.0_{\pm 12.8}$
	SEALS	3.2k	$95.3_{\pm 1.6}$	$69.2_{\pm 14.7}$	$83.8_{\pm 2.7}$	$93.2_{\pm 1.6}$	$67.8_{\pm 12.6}$
	No-Op	475	$10.8_{\pm 0.3}$	$4.7_{\pm 3.2}$	6h	$10.8_{\pm 0.3}$	$4.7_{\pm 3.2}$
WikiToxic	AnchorAL	4.2k	$128.1_{\pm 0.6}$	$119.0_{\pm 1.2}$	$2.6_{\pm 0.0}$	$106.6_{\pm 0.5}$	$99.0_{\pm 1.1}$
	RandSub	4.0k	$119.4_{\pm 0.2}$	$107.4_{\pm 0.8}$	$20.4_{\pm 0.8}$	$104.7_{\pm 0.4}$	$94.6_{\pm 1.4}$
	SEALS	3.6k	$109.6_{\pm 0.7}$	$100.6_{\pm 1.8}$	$60.9_{\pm 1.5}$	$105.9_{\pm 0.5}$	$97.6_{\pm 1.3}$
	No-Op	2.9k	$88.3_{\pm 0.4}$	$80.8_{\pm 1.0}$	6h	$88.3_{\pm 0.4}$	$80.8_{\pm 1.0}$
Agnews-Bus	AnchorAL	5k	$179.2_{\pm 0.1}$	$118.2_{\pm 1.3}$	$2.7_{\pm 0.1}$	*same as Overall*	
	RandSub	5k	$179.2_{\pm 0.4}$	$118.0_{\pm 4.0}$	$21.5_{\pm 0.1}$		
	SEALS	5k	$179.6_{\pm 0.1}$	$120.8_{\pm 0.8}$	$62.2_{\pm 1.3}$		

Table 2: AUC of the F1-score, total annotated budget, and total instance selection time for the BERT-base model and Entropy AL strategy. Median and interquartile range across 8 runs.

AL Strategies. We use one AL strategy from each type of AL approach (Dasgupta, 2011). Entropy (Joshi et al., 2009, uncertainty-based) selects instances with the highest predictive entropy. FT-BERTKM (Yuan et al., 2020, diversity-based) chooses instances nearest to cluster centres obtained running k-MEANS clustering with k equal to the number of instances to label. Finally, BADGE (Ash et al., 2020, hybrid) selects instances using the k-MEANS++ initialisation scheme applied to the gradient embeddings. More details in App. A.

Models. We consider 6 models of different sizes and architectures (Table 1). Following the transformers library (Wolf et al., 2020) implementation, we add a linear layer to the model representations: the [CLS] token for encoders, the last non-padding token for decoders, and the end-of-sequence token for encoder-decoders. Training details in App. A.

Datasets. We consider 4 text classification tasks, both binary and multiclass. Using the AmazonCat-13k dataset (McAuley and Leskovec, 2013) we construct the Amazon-Agri task, selecting the "agriculture" (0.09%) category as target, and the Amazon-Multi task which also has "archaeology" (0.09%), "audio" (0.56%), and "philosophy" (0.78%). Moreover, we consider the WikiToxic (Adams et al., 2017) and Agnews-Bus—i.e., Agnews (Zhang et al., 2015) binarised using the "business" topic as the target—tasks. We make them imbalanced downsampling their minority class to 1%. More details in App. B.

5 Results

Table 2 summarises a subset of our results for the BERT-base model and the Entropy strategy. The full results table in App. C provides similar insights for the other models and AL strategies. Since we limit the duration of each run to 6h, slow pool filtering strategies might not be able to execute all iterations within the time budget. Thus, we report metrics at the end of the 6h time budget ("Overall") and at the biggest common iteration completed by all methods ("Budget-Matched").[11] For each experiment, we report the total annotated budget within the 6h limit ("Budget"), the AUC of the (macro-averaged) F1-score on the majority ("Majority") and minority ("Minority") classes, and the total instances selection time ("Time").

Cost Efficiency. Broadly, the overall annotation cost can be divided into the cost of the compute needed to run inference on the pool and the cost of the human annotation. Pool filtering methods reduce both cost components by considering only a subset of the pool in each iteration: fewer instances require less compute and are faster to process which makes annotators spend less time waiting for instances to label resulting in more annotations within the same time budget. AnchorAL is the

[11] For example, if SEALS only completes $\hat{t}$ steps within the available time while AnchorAL manages to complete all iterations, the Budget-Matched performance compares the results after $\hat{t} \times b$ instances are annotated.

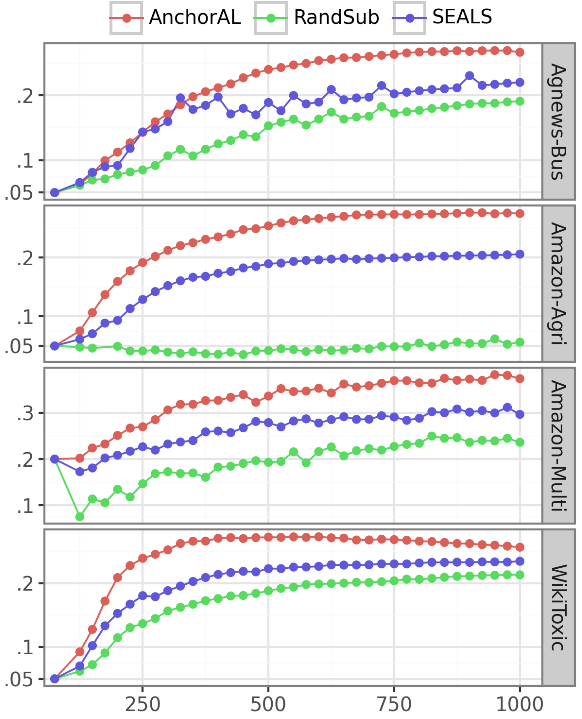

Figure 3: Proportion of minority instances (y-axis) discovered within 1k annotations (x-axis). Geometric average across all AL strategies considered for BERT-base.

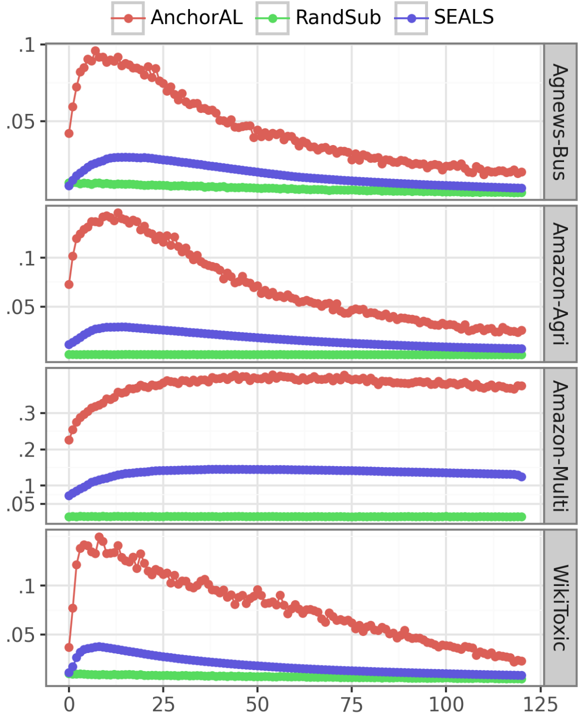

Figure 4: Proportion of minority instances (y-axis) in the subpool at each iteration (x-axis). Geometric average across all AL strategies considered for BERT-base.

fastest method overall, reducing the total annotators' waiting time ("Time" column) from hours to less than 5 minutes and labelling the most instances within the time budget ("Budget" column).

AnchorAL is faster because it returns a smaller subpool than the other methods (Fig. 2 panel a). For example, AnchorAL returns $<1\text{k}$ instances at each iteration for the binary tasks while in SEALS, by design, the subpool grows across iterations and for RandSub a larger subpool is required to achieve good performance. In theory, RandSub can be made arbitrarily fast by choosing a small $\bar{M}$ but it comes with high performance costs. For example, setting $\bar{M} = 1\text{k}$, similar to AnchorAL, results in a dramatic performance drop (Table 3 last row). In the limiting case of setting $\bar{M} = b$ RandSub reduces to Random.

Performance. AnchorAL is the best-performing method for both minority and majority classes per time- ("Overall" columns) and budget-unit ("Budget-Matched" columns) across models and AL strategies. The only exception is Agnews-Bus where it slightly underperforms SEALS on the minority class(es). We hypothesise that this might be an artefact of the downsampling procedure we applied to this task to make it imbalanced: minority instances may be near the initial decision boundary, making exploration of the input space detrimental. Furthermore, we notice that RandSub is a stronger baseline than reported by Coleman et al. (2022). We hypothesise that this discrepancy depends on the fact that we train the entire model rather than only the classification head.

Discovery of Minority Instances. Although our primary goal in applying AL is to induce a good predictive model, we also report the number of minority instances discovered by each method in Fig. 3, which is likely to be indicative of the usefulness of the annotations when re-used for training other models. At every iteration, AnchorAL discovers more minority instances than the baselines resulting in a more balanced labelled set.

Contributions

AnchorAL is the fastest method, reducing the total selection time from hours to minutes, thus allowing for an interactive annotation setup. It is (often) the best-performing, reaching higher performance in less time and with fewer annotations. Finally, it discovers the most minority instances resulting in more balanced labelled sets.

$\Gamma_{\texttt{maj}}$	$\Gamma_{\texttt{min}}$	A	K	**Maj.**	**Min.**	**Time**
kM++	kM++	10	50	134.5	73.5	4.1
Ent	kM++	10	50	134.5	71.5	12.7
Ent	Ent	10	50	134.5	71.4	12.8
kM++	kM++	50	50	134.5	72.6	14.7
kM++	kM++	100	50	134.5	72.2	24.8
kM++	kM++	10	500	134.5	70.7	30.1
kM++	kM++	10	5k	134.5	68.3	39.0
No anchoring				133.7	33.6	3.4

Table 3: Effect of hyper-parameters on AnchorAL's performance (F1-score) for the BERT-base model on the Amazon-Agri task using the Entropy AL strategy. Defaults in the top row, changes highlighted in grey.

6 Analysis

In this section, we study AnchorAL's anchoring mechanism. This mechanism is the core component allowing AnchorAL to reach higher performance despite using a fixed-sized, small subpool. Everything else being equal, if the anchoring mechanism is turned off the performance is dramatically affected, as shown in the last row of Table 3 which reports the performance of AnchorAL with no anchoring. We analyse the composition of the subpool at each iteration and test the effects of different hyper-parameters settings on performance by experimenting with the BERT-base model on the Amazon-Agri dataset using the Entropy strategy.

6.1 Subpool Composition

The anchoring mechanism determines the composition of the subpool. In Fig. 4 we report the proportion of minority instances in the subpool returned by the methods considered. We observe that, across iterations, AnchorAL consistently returns subpools with more minority instances; this results in more balanced labelled sets, as shown in Fig. 3, and ultimately better performance. Therefore, the key intuition in AnchorAL is to return a more balanced subpool which allows any AL strategy to discover and select minority instances more easily, without the need for a large subpool (e.g., like RandSub) which reduces the instance selection time.

6.2 Ablations

The anchoring mechanism is controlled by three hyperparameters: the number of anchors selected per class (A), the anchor selection strategy (Γ), and the number of neighbours retrieved from the subpool per anchor (K). Together, A and K control the size of the resulting subpool while Γ determines which anchors are selected and, thus, which part of the input space to explore. In §3 we presented our default settings; here we motivate their choice.

Anchor Selection Strategy (Γ). We experiment with different anchor selection strategies for both majority ($\Gamma_{\texttt{maj}}$) and minority ($\Gamma_{\texttt{min}}$) anchors (Table 3 row 2-3). As the alternative strategy, we use the entropy of the model. The reasoning is as follows: since the model knows the majority better than the minority class(es) it might be informative in choosing majority instances near the decision boundary. Overall, we report a negative effect, even more pronounced when entropy is used to select minority anchors too. We hypothesise that using a model-agnostic anchors selection strategy (e.g., k-MEANS++) avoids propagating the initial biases of the model in the selection of the instances.

Number of Anchors (A). We vary the number of anchors from 10 to 50 and 100 (Table 3 row 4-5). First, note that using all the labelled data as anchors at each iteration becomes quickly impractical (e.g., like SEALS); since only a few iterations are completed within the 6h limit, the performance is almost zero and we omit it. Second, there is a negative correlation between the number of anchors and performance. We hypothesise this is due to the bigger and more imbalanced resulting subpool which decreases the benefits of AnchorAL.

Number of Neighbours (K). We change the default number of neighbours from 50 to 500 and 5k (Table 3 row 6-7). Performance degrades as we retrieve more neighbours from the pool. Moreover, as the resulting subpool is bigger, instance selection time increases even though it plateaus after 500 as the anchors retrieve the same instances that are then aggregated (Alg. 1-Line 8).

7 Conclusions

We propose AnchorAL, a novel pool filtering method designed to scale AL to large pools while addressing class imbalance. AnchorAL uses the semantic representation capabilities of language models to explore the input space and create a fixed-sized, smaller, more balanced, and different subpool in each iteration. By running the AL strategy on the subpool, AnchorAL promotes the discovery of minority instances, prevents overfitting to the initial labelled set, and obtains a constant instance selection time, independently of the original pool size.

Limitations

In this section, we discuss the limitations of the scope of this work and the threats to the external validity of our results.

Languages. We experimented with a limited number of languages (only English). We do not have experimental evidence that our method can work for other languages for which good embedding models are not available. Still, our approach has been built without any language-specific constraints or resources. Our method can be applied to any other language for which these resources are available.

Realism. Recent AL research emphasises the empirical evaluation of classifier performance resulting from simulated experiments. However, this idealised setting tacitly makes assumptions that cannot be true in real-world settings (Margatina and Aletras, 2023); for example, a perfect oracle, uniform annotation difficulty, and the possibility to monitor the performance of the AL strategy on a test set while training (Wallace et al., 2010; Levonian et al., 2022). Our paper suffers from these limitations too, even though we strived to address the annotators' waiting time issue. We leave for future work exploring methods to make AnchorAL more suited for practical use in real-world annotation settings.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 Research and Innovation programme grant AVeriTeC (Grant agreement No. 865958). We thank the anonymous reviewers for their helpful questions and comments that helped us improve the paper. We thank Tiago Pimentel, Davide Lesci, and Marco Lesci for their help in proofreading the final version of the paper.

References

C.J. Adams, Jeffrey Sorensen, Julia Elliott, Lucas Dixon, Mark McDonald, Nithum Thain, and Will Cukierski. 2017. Toxic comment classification challenge.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 233–242. PMLR.

David Arthur and Sergei Vassilvitskii. 2007. K-Means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, USA. Society for Industrial and Applied Mathematics.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2020. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*.

Les Atlas, David Cohn, and Richard Ladner. 1989. Training connectionist networks with queries and selective sampling. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Josh Attenberg and Şeyda Ertekin. 2013. Class imbalance and active learning. In *Imbalanced Learning*, chapter 6, pages 101–149. John Wiley & Sons, Ltd.

Josh Attenberg and Foster Provost. 2010. Why label when you can search?: Alternatives to active learning for applying human resources to build classification models under extreme class imbalance. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 423–432, Washington DC USA. ACM.

Artem Babenko and Victor Lempitsky. 2016. Efficient indexing of billion-scale datasets of deep descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2055–2063.

Maria-Florina Balcan, Alina Beygelzimer, and John Langford. 2009. Agnostic active learning. *Journal of Computer and System Sciences*, 75(1):78–89.

Maria Florina Balcan and Steve Hanneke. 2012. Robust interactive learning. In *Proceedings of the 25th Annual Conference on Learning Theory*, pages 20.1–20.34. JMLR Workshop and Conference Proceedings.

Maria-Florina Balcan, Steve Hanneke, and Jennifer Wortman Vaughan. 2010. The true sample complexity of active learning. *Machine Learning*, 80(2):111–139.

Yoram Baram, Ran El Yaniv, and Kobi Luz. 2004. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5(Mar):255–291.

Alina Beygelzimer, Daniel J Hsu, John Langford, and Chicheng Zhang. 2016. Search improves label for active learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2022. On the opportunities and risks of foundation models.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. SMOTE: Synthetic minority oversampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. 2021. Batch active learning at scale. In *Advances in Neural Information Processing Systems*, volume 34, pages 11933–11944. Curran Associates, Inc.

Cody Coleman, Edward Chou, Julian Katz-Samuels, Sean Culatana, Peter Bailis, Alexander C. Berg, Robert Nowak, Roshan Sumbaly, Matei Zaharia, and I. Zeki Yalniz. 2022. Similarity search for efficient active learning and search of rare concepts. In *Proceedings of the First MiniCon Conference*.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2020. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9268–9277.

Sanjoy Dasgupta. 2011. Two faces of active learning. *Theoretical Computer Science*, 412(19):1767–1781.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dmitriy Dligach and Martha Palmer. 2011. Good seed makes a good crop: Accelerating active learning using language modeling. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 6–10, Portland, Oregon, USA. Association for Computational Linguistics.

Qi Dong, Shaogang Gong, and Xiatian Zhu. 2017. Class rectification hard mining for imbalanced deep learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1851–1860.

Seyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. 2007. Learning on the border: Active learning in imbalanced data classification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 127–136, Lisbon Portugal. ACM.

Roman Garnett, Yamuna Krishnamurthy, Xuehan Xiong, Jeff Schneider, and Richard Mann. 2012. Bayesian optimal active search and surveying. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 843–850, Madison, WI, USA. Omnipress.

Daniel Grießhaber, Johannes Maucher, and Ngoc Thang Vu. 2020. Fine-tuning BERT for low-resource natural language understanding via active learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1158–1171, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Suchin Gururangan, Dallas Card, Sarah Dreier, Emily Gade, Leroy Wang, Zeyu Wang, Luke Zettlemoyer, and Noah A. Smith. 2022. Whose language counts as high quality? Measuring language ideologies in text data selection. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2562–2580, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Steve Hanneke. 2007. A bound on the label complexity of agnostic active learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 353–360, New York, NY, USA. Association for Computing Machinery.

Steve Hanneke. 2011. Rates of convergence in active learning. *The Annals of Statistics*, 39(1):333–361.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature*, 585(7825):357–362.

Jason S Hartford, Kevin Leyton-Brown, Hadas Raviv, Dan Padnos, Shahar Lev, and Barak Lenz. 2020. Exemplar guided active learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 13163–13173. Curran Associates, Inc.

Haibo He and Edwardo A. Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Sophie Henning, William Beluch, Alexander Fraser, and Annemarie Friedrich. 2023. A survey of methods for addressing class imbalance in deep-learning based natural language processing. In *European Chapter of the Association for Computational Linguistics (EACL 2023)*.

Shali Jiang, Roman Garnett, and Benjamin Moseley. 2019. Cost effective active search. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Shali Jiang, Gustavo Malkomes, Matthew Abbott, Benjamin Moseley, and Roman Garnett. 2018. Efficient nonmyopic batch active search. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs.

Justin M. Johnson and Taghi M. Khoshgoftaar. 2019. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):27.

Ajay J. Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. 2009. Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379.

Jaeyoung Kim, Dongbin Na, Sungchul Choi, and Sungbin Lim. 2023. Bag of tricks for in-distribution calibration of pretrained transformers. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 551–563, Dubrovnik, Croatia. Association for Computational Linguistics.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. 2019. BatchBALD: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland. Association for Computational Linguistics.

Zachary Levonian, Chia-Jung Lee, Vanessa Murdock, and F. Maxwell Harper. 2022. Trade-offs in sampling and search for early-stage interactive text classification. In *27th International Conference on Intelligent User Interfaces*, IUI '22, pages 566–583, New York, NY, USA. Association for Computing Machinery.

Christopher Lin, Mausam Mausam, and Daniel Weld. 2018. Active learning with unbalanced classes and example-generation queries. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 6:98–107.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A. Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Seiji Maekawa, Dan Zhang, Hannah Kim, Sajjadur Rahman, and Estevam Hruschka. 2022. Low-resource interactive active labeling for fine-tuning language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3230–3242, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yu A. Malkov and D. A. Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836.

Katerina Margatina and Nikolaos Aletras. 2023. On the limitations of simulating active learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4402–4419, Toronto, Canada. Association for Computational Linguistics.

Katerina Margatina, Loic Barrault, and Nikolaos Aletras. 2022. On the importance of effectively adapting pretrained language models for active learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 825–836, Dublin, Ireland. Association for Computational Linguistics.

Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. 2021. Active learning by acquiring contrastive examples. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 650–663, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Christoph Mayer and Radu Timofte. 2020. Adversarial sampling for active learning. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 3060–3068, Snowmass Village, CO, USA. IEEE.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 165–172, New York, NY, USA. Association for Computing Machinery.

Sankha Subhra Mullick, Shounak Datta, and Swagatam Das. 2019. Generative adversarial minority oversampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1695–1704.

T. Osugi, Deng Kim, and S. Scott. 2005. Balancing exploration and exploitation: A new algorithm for active machine learning. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–16.

Seulki Park, Jongin Lim, Younghan Jeon, and Jin Young Choi. 2021. Influence-balanced loss for imbalanced visual classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 735–744.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato. 2019. Bayesian batch active learning as sparse subset approximation. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Christopher Schröder, Andreas Niekler, and Martin Potthast. 2022. Revisiting uncertainty-based query strategies for active learning with transformers. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2194–2203, Dublin, Ireland. Association for Computational Linguistics.

Hinrich Schütze, Emre Velipasaoglu, and Jan O. Pedersen. 2006. Performance thresholding in practical text classification. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 662–671, New York, NY, USA. Association for Computing Machinery.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*.

Burr Settles. 2012. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Springer International Publishing, Cham.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MPNet: Masked and permuted pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 33, pages 16857–16867. Curran Associates, Inc.

H. Steinhaus. 1956. Sur la division des corps matériels en parties. *Bull. Acad. Polon. Sci. Cl. III.*, 4:801–804.

Srikanth Thudumu, Philip Branch, Jiong Jin, and Jugdutt (Jack) Singh. 2020. A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, 7(1):42.

Katrin Tomanek, Florian Laws, Udo Hahn, and Hinrich Schütze. 2009. On proper unit selection in Active Learning: Co-selection effects for named entity recognition. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language*

Processing, pages 9–17, Boulder, Colorado. Association for Computational Linguistics.

Akim Tsvigun, Artem Shelmanov, Gleb Kuzmin, Leonid Sanochkin, Daniil Larionov, Gleb Gusev, Manvel Avetisian, and Leonid Zhukov. 2022. Towards computationally feasible deep active learning. In *Findings 2022*, pages 1198–1218, Seattle, United States. Association for Computational Linguistics.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models.

Jason Van Hulse, Taghi M. Khoshgoftaar, and Amri Napolitano. 2007. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 935–942, New York, NY, USA. Association for Computing Machinery.

Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Byron C. Wallace, Kevin Small, Carla E. Brodley, and Thomas A. Trikalinos. 2010. Active learning for biomedical citation screening. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 173–182, Washington DC USA. ACM.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. 2017. Learning to model the tail. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Donggeun Yoo and In So Kweon. 2019. Learning loss for active learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 93–102, Long Beach, CA, USA. IEEE.

Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2019. AttentionXML: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. 2020. Cold-start active learning through self-supervised language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7935–7948, Online. Association for Computational Linguistics.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2021. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

A Experimental Details

In addition to the information already present in §4, here we add more specific implementation details.

A.1 Index Construction

We embed each document using the MPNet encoder (Song et al., 2020)[12] available on the HuggingFace Hub and implemented through the Sentence-Transformers[13] library (Reimers and Gurevych, 2019). Embedding a batch of 1,024 documents of size on our hardware (App. A.5) takes circa 5.5 seconds. For efficient dense retrieval, we use Hierarchical Navigable Small World (HNSW) approximate nearest neighbour search algorithm proposed by Malkov and Yashunin (2018) and available through the hnswlib[14] library. When building the index we use cosine similarity as the distance metric and the following standard settings: `ef_construction` = 200, `ef` = 200, and `M` = 64 to control the speed/accuracy trade-off during the index construction, query time/accuracy trade-off, and the maximum number of outgoing connections in the graph, respectively.

A.2 Training Details

We use the AdamW optimiser (Loshchilov and Hutter, 2018) with the default hyperparameters—as implemented in Pytorch, that is $\beta_1 = .9$, $\beta_2 = .999$, `weight_decay` $= .01$, `eps` $= 10^{-8}$—and set the learning rate to 4×10^{-5} (2×10^{-4} for BERT-tiny) with a constant schedule and a batch size of 32. We truncate the input sequences to 512 tokens. We do not assume access to a validation set and, instead, train for 10 epochs selecting the best model based on the training F1-score on the minority class(es). Since the number of batches in the initial iterations is limited ($3 \approx 100/32$), to allow the model to converge we set the minimum number of optimisation steps to 100 and use early stopping with a minimum delta of 10^{-5}.

A.3 Reproducibility

We implement all experiments using the PyTorch (Paszke et al., 2019) framework. We assign a different pseudo-random number generator to data shuffling, model initialisation, AL strategy, and pool subsampling strategy. In addition, we use CUDA deterministic operations and set the seeds for pseudo-random number generators in PyTorch, Numpy (Harris et al., 2020), the Python Random module (Van Rossum and Drake, 2009), and for each multi-processing worker.

A.4 Active Learning Strategies

In this section we provide an overview of the AL strategy used in the paper.

Entropy. The Entropy strategy scores instances by computing the predictive entropy of the probability distribution assigned by the model, that is

$$H(\boldsymbol{x}) = -\sum_{c=1}^{C} f(\boldsymbol{x})_c \log f(\boldsymbol{x})_c$$

FT-BERTKM. The FT-BERTKM strategy computes instance representations using the trained model. These representations correspond to the output of the penultimate layer of the model $h(\boldsymbol{x})$, which is the input to the classification layer. Once these representations are computed for each unlabelled instance and $l2$-normalised, it runs k-MEANS setting k equal to the number of instances to select in each iteration. Finally, the cluster centres (or the closest instances in the embedding space) are selected.

BADGE. The goal of BADGE is to sample a diverse and uncertain batch of points for training neural networks. The algorithm transforms data into representations that encode model confidence and then clusters these transformed points. First, an instance $\boldsymbol{x}$ is passed through the trained model to obtain its predicted label

$$\hat{y} = \operatorname{argmax} f(\boldsymbol{x})$$

Next, a gradient embedding of the last layer of the model $g_{\boldsymbol{x}}$ is obtained with respect to the loss computed using the predicted labels as the target, that is

$$g_{\boldsymbol{x}} = \nabla_\theta \mathcal{L}\left(f_\theta(\boldsymbol{x}), \hat{y}\right)$$

where $\mathcal{L}$ is the cross-entropy loss and θ are parameters of the last layer of the model. The gradient embeddings can be computed in closed form as

$$(g_{\boldsymbol{x}})_c = \left[f_\theta(\boldsymbol{x})_c - \mathbb{1}(\hat{y} = c)\right] \; h_\theta(\boldsymbol{x})$$

where $h(\cdot)$ computes the model representation that feeds into the last layer. The gradient embedding is a multi-dimensional tensor since we are computing the gradient with respect to all possible classes

[12] huggingface.co/sentence-transformers/all-mpnet-base-v2.

[13] sbert.net.

[14] github.com/nmslib/hnswlib.

$c \in \{1, ..., C\}$. Thus, the c-th block of $g_{\boldsymbol{x}}$ is the hidden representation $h(\boldsymbol{x})$ scaled by the difference between model confidence score $f(\boldsymbol{x})_c$ and an indicator function that indicates whether the predicted class $\hat{y}$ is label c. Finally, BADGE chooses a batch to sample by applying k-MEANS++ on these (flattened) gradient embeddings. These embeddings consist of model confidence scores and hidden representations, so they encode information about both uncertainty and the data distribution. By applying k-MEANS++ on the gradient embeddings, the chosen examples differ in feature representation and predictive uncertainty.

A.5 Hardware Details

We use a server with one NVIDIA A100 80GB PCIe, 32 CPUs, and 32 GB of RAM for all experiments. Below, we report a subset of the output of the lscpu command:

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Address sizes:         46 bits physical,
                       48 bits virtual
Byte Order:            Little Endian
CPU(s):                32
On-line CPU(s) list:   0-31
Vendor ID:             GenuineIntel
Model name:            Intel(R) Xeon(R)
                       Silver 4210R CPU
                       @ 2.40GHz
CPU family:            6
Model:                 85
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             8
Stepping:              7
BogoMIPS:              4800.11
```

B Data

Our data pipeline is divided into two steps: sourcing and preparation.

B.1 Data Sourcing

We download the raw datasets from their original sources or other comparable faithful source (e.g., HuggingFace Hub only when the dataset is created by the original authors or the HuggingFace team).

AmazonCat-13k. The AmazonCat-13k dataset was released by McAuley and Leskovec (2013) and is composed of product descriptions and reviews classified into 13k multi-label categories. The dataset is split into 1.2M train and 300k evaluation instances. It is commonly used as an extreme classification benchmark (You et al., 2019)[15] where the goal is to classify an item into its categories. We download the raw data from the original data source.[16]

		Test		Train	
Dataset	**Label**	**#**	**%**	**#**	**%**
Amazon-Agri	Negative	5,000	94.61%	1,185,188	99.91%
	Positive	285	5.39%	1,051	0.09%
		5,285	100%	1,186,239	100%
Amazon-Multi	agriculture	285	2.92%	1,051	0.09%
	archaeology	256	2.62%	1,010	0.09%
	audio	1,759	18.03%	6,651	0.56%
	others	5,000	51.26%	1,168,266	98.48%
	philosophy	2,454	25.16%	9,261	0.78%
		9,754	100%	1,186,239	100%
Agnews-Bus	Negative	5,700	75.00%	90,000	99.00%
	Positive	1,900	25.00%	910	1.00%
		7,600	100%	90,910	100%
WikiToxic	non	5,000	44.47%	114,722	99.00%
	tox	6,243	55.53%	1,159	1.00%
		11,243	100%	115,881	100%

Table 4: Data statistics of the task considered.

Agnews. The AG dataset[17] consists of more than 1M news articles written in English. The articles have been collected from more than 2k news sources by ComeToMyHead, an academic news search engine running since July 2004, over a period of more than one year. The Agnews topic classification dataset is a curated version proposed in Zhang et al. (2015) and is constructed by choosing the 4 largest classes from the original corpus: World, Sports, Business, Sci/Tech. The dataset is split into 120k train and 7.6k evaluation instances. We use the version available on the HuggingFace Hub.[18]

WikiToxic. The WikiToxic dataset is an updated version of the Kaggle Toxic Comment dataset used in the homonymous 2017/2018 challenge.[19] It contains comments in English collected from Wikipedia forums and classifies them into two categories, Toxic and Non-toxic. The dataset is split into 128k train, 32k validation, and 64k test instances. We download the data from the HuggingFace Hub.[20]

[15] manikvarma.org/downloads/XC/XMLRepository.html.

[16] drive.google.com/u/0/uc?id=17rVRDarPwlMpb3l5zof9h34FlwbpTu4l.

[17] groups.di.unipi.it/ gulli/AG_corpus_of_news_articles.html.

[18] huggingface.co/datasets/ag_news.

[19] kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/overview.

[20] huggingface.co/datasets/OxAISH-AL-LLM/wiki_toxic.

B.2 Data Preparation

Data preparation refers to the process of converting a dataset into a task, that is preparing the data for training and evaluation which entails encoding the labels to integers and tokenising the documents. First, we only use the respective training and test sets for all the datasets since we do not assume access to a fixed validation set when training the model in each iteration. Second, we tokenise the documents using the model-specific tokeniser available in the tokenizers[21] library (Wolf et al., 2020). Third, we encode the labels by mapping each class to an integer. Finally, we optionally downsample the minority class to increase the imbalance. Below we report the process of creating the classification tasks we use in our experiments from the original datasets. In Table 4, we report statistics about the *prepared* datasets used in our experiments.

Since AmazonCat-13k is a multilabel dataset, we choose some of the labels and create binary and multiclass classification tasks. We create one binary classification task, Amazon-Agri, by choosing labels referring to the "agricultural sciences" topic: 246. If any of the topic-specific labels appear in the label set of a document we assign the document to the minority class. We create the multiclass task, Amazon-Multi, by using the "agriculture" label; the "philosophy" label corresponding to the indices 413, 3999, 4044, 5881, 8900, 8901, 10990, 13319; the "archaeology" label with indices 538, 539, 8601; and the "audio" label with index 677.

For Agnews we choose a label and treat it as the minority class and we downsample it in the training set to increase the imbalance. We choose the "Business" class and create the Agnews-Bus task. We make the minority class account for only 1% of the training set.

Finally, for WikiToxic, being already a binary classification task, we only downsample the "tox" class to only account for 1% of the training set.

C Full Table of Results

The table follows in the next pages and reports the AUC of the F1-score, total annotated budget, and total instance selection time for the BERT-base model and Entropy AL strategy. Median and interquartile range across 8 runs. Results at the end of the $6\mathrm{h}$ budget for each run (Overall) and at the biggest common annotated budget by all pool filtering strategies (Budget-Matched) differences in budget highlighted in grey.

[21] github.com/huggingface/tokenizers.

				Overall				Budget-Matched			
Dataset	**Model**	**AL Strategy**	**Pool Filtering**	**Budget**	**Majority**	**Minority**	**Time**	**Budget**	**Majority**	**Minority**	**Time**
Agnews-Bus	ALBERT-base	Entropy	AnchorAL	5.0k	$176.5_{\pm 0.7}$	$96.9_{\pm 10.0}$	$2.7_{\pm 0.1}$	5.0k	$176.5_{\pm 0.7}$	$96.9_{\pm 10.0}$	$2.7_{\pm 0.1}$
			RandSub	5.0k	$176.7_{\pm 1.8}$	$98.3_{\pm 5.5}$	$22.4_{\pm 0.5}$	5.0k	$176.7_{\pm 1.8}$	$98.3_{\pm 5.5}$	$22.4_{\pm 0.5}$
			SEALS	5.0k	$176.3_{\pm 1.2}$	$98.3_{\pm 4.1}$	$66.5_{\pm 1.2}$	5.0k	$176.3_{\pm 1.2}$	$98.3_{\pm 4.1}$	$66.5_{\pm 1.2}$
	BERT-base	BADGE	AnchorAL	5.0k	$179.1_{\pm 0.3}$	$116.9_{\pm 2.3}$	$2.9_{\pm 0.1}$	5.0k	$179.1_{\pm 0.3}$	$116.9_{\pm 2.3}$	$2.9_{\pm 0.1}$
			RandSub	5.0k	$179.3_{\pm 0.4}$	$118.8_{\pm 2.8}$	$23.7_{\pm 0.1}$	5.0k	$179.3_{\pm 0.4}$	$118.8_{\pm 2.8}$	$23.7_{\pm 0.1}$
			SEALS	5.0k	$179.6_{\pm 0.3}$	$121.1_{\pm 2.9}$	$74.0_{\pm 1.4}$	5.0k	$179.6_{\pm 0.3}$	$121.1_{\pm 2.9}$	$74.0_{\pm 1.4}$
		Entropy	AnchorAL	5.0k	$179.2_{\pm 0.1}$	$118.2_{\pm 1.3}$	$2.7_{\pm 0.1}$	5.0k	$179.2_{\pm 0.1}$	$118.2_{\pm 1.3}$	$2.7_{\pm 0.1}$
			RandSub	5.0k	$179.2_{\pm 0.4}$	$118.0_{\pm 4.0}$	$21.5_{\pm 0.1}$	5.0k	$179.2_{\pm 0.4}$	$118.0_{\pm 4.0}$	$21.5_{\pm 0.1}$
			SEALS	5.0k	$179.6_{\pm 0.1}$	$120.8_{\pm 0.8}$	$62.2_{\pm 1.3}$	5.0k	$179.6_{\pm 0.1}$	$120.8_{\pm 0.8}$	$62.2_{\pm 1.3}$
		FT-BERTKM	AnchorAL	5.0k	$178.2_{\pm 0.5}$	$110.3_{\pm 5.4}$	$2.9_{\pm 0.1}$	5.0k	$178.2_{\pm 0.5}$	$110.3_{\pm 5.4}$	$2.9_{\pm 0.1}$
			RandSub	5.0k	$176.6_{\pm 0.3}$	$95.9_{\pm 3.4}$	$25.1_{\pm 0.2}$	5.0k	$176.6_{\pm 0.3}$	$95.9_{\pm 3.4}$	$25.1_{\pm 0.2}$
			SEALS	5.0k	$176.5_{\pm 0.6}$	$96.2_{\pm 6.9}$	$84.4_{\pm 4.6}$	5.0k	$176.5_{\pm 0.6}$	$96.2_{\pm 6.9}$	$84.4_{\pm 4.6}$
		Random	No-Op	5.0k	$172.4_{\pm 1.2}$	$59.4_{\pm 12.9}$	$0.0_{\pm 0.0}$	5.0k	$172.4_{\pm 1.2}$	$59.4_{\pm 12.9}$	$0.0_{\pm 0.0}$
	BERT-tiny	Entropy	AnchorAL	5.0k	$172.6_{\pm 1.2}$	$54.3_{\pm 14.4}$	$1.4_{\pm 0.0}$	5.0k	$172.6_{\pm 1.2}$	$54.3_{\pm 14.4}$	$1.4_{\pm 0.0}$
			RandSub	5.0k	$168.3_{\pm 0.7}$	$4.0_{\pm 9.6}$	$7.5_{\pm 0.1}$	5.0k	$168.3_{\pm 0.7}$	$4.0_{\pm 9.6}$	$7.5_{\pm 0.1}$
			SEALS	5.0k	$168.1_{\pm 0.3}$	$1.6_{\pm 3.8}$	$26.3_{\pm 0.6}$	5.0k	$168.1_{\pm 0.3}$	$1.6_{\pm 3.8}$	$26.3_{\pm 0.6}$
	DeBERTa-base	Entropy	AnchorAL	3.5k	$125.3_{\pm 0.3}$	$75.8_{\pm 2.4}$	$2.3_{\pm 0.0}$	2.9k	$102.4_{\pm 0.5}$	$60.8_{\pm 3.6}$	$1.9_{\pm 0.1}$
			RandSub	3.1k	$110.8_{\pm 0.1}$	$66.0_{\pm 1.1}$	$21.0_{\pm 0.3}$	2.9k	$102.7_{\pm 0.3}$	$61.5_{\pm 2.7}$	$19.4_{\pm 0.2}$
			SEALS	2.9k	$102.6_{\pm 0.2}$	$60.5_{\pm 2.0}$	$40.1_{\pm 0.7}$	2.9k	$102.6_{\pm 0.2}$	$60.5_{\pm 2.0}$	$40.1_{\pm 0.7}$
	GPT-2	Entropy	AnchorAL	5.0k	$168.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$2.6_{\pm 0.0}$	4.7k	$158.6_{\pm 0.0}$	$0.0_{\pm 0.0}$	$2.5_{\pm 0.0}$
			RandSub	5.0k	$168.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$20.0_{\pm 0.1}$	4.7k	$158.6_{\pm 0.0}$	$0.0_{\pm 0.0}$	$19.0_{\pm 0.1}$
			SEALS	4.7k	$158.6_{\pm 0.0}$	$0.0_{\pm 0.0}$	$73.4_{\pm 0.7}$	4.7k	$158.6_{\pm 0.0}$	$0.0_{\pm 0.0}$	$73.4_{\pm 0.7}$
	T5-base	Entropy	AnchorAL	3.7k	$131.6_{\pm 0.2}$	$77.9_{\pm 2.0}$	$4.0_{\pm 0.0}$	3.3k	$116.2_{\pm 0.3}$	$69.1_{\pm 2.4}$	$3.5_{\pm 0.1}$
			RandSub	3.6k	$128.3_{\pm 0.1}$	$69.5_{\pm 1.3}$	$37.5_{\pm 1.6}$	3.3k	$115.4_{\pm 0.3}$	$61.2_{\pm 2.3}$	$35.2_{\pm 0.2}$
			SEALS	3.3k	$115.2_{\pm 0.3}$	$58.9_{\pm 3.0}$	$96.3_{\pm 6.5}$	3.3k	$115.2_{\pm 0.3}$	$58.9_{\pm 3.0}$	$96.3_{\pm 6.5}$
Amazon-Agri	ALBERT-base	Entropy	AnchorAL	3.6k	$134.2_{\pm 0.7}$	$46.7_{\pm 2.9}$	$4.9_{\pm 0.1}$	3.1k	$115.9_{\pm 0.9}$	$43.5_{\pm 1.4}$	$4.1_{\pm 0.1}$
			RandSub	3.4k	$129.6_{\pm 0.5}$	$27.5_{\pm 16.7}$	$31.2_{\pm 0.0}$	3.1k	$116.0_{\pm 0.4}$	$25.5_{\pm 13.4}$	$27.9_{\pm 0.0}$
			SEALS	3.1k	$116.4_{\pm 0.3}$	$45.4_{\pm 2.2}$	$93.1_{\pm 1.2}$	3.1k	$116.4_{\pm 0.3}$	$45.4_{\pm 2.2}$	$93.1_{\pm 1.2}$
	BERT-base	BADGE	AnchorAL	3.7k	$142.4_{\pm 0.1}$	$76.3_{\pm 3.3}$	$4.5_{\pm 0.1}$	3.1k	$119.8_{\pm 0.1}$	$63.5_{\pm 2.8}$	$3.8_{\pm 0.1}$
			No-Op	425	$12.7_{\pm 0.0}$	$2.6_{\pm 1.0}$	6h	425	$12.7_{\pm 0.0}$	$2.6_{\pm 1.0}$	6h
			RandSub	3.6k	$138.2_{\pm 0.1}$	$60.8_{\pm 2.0}$	$30.5_{\pm 0.4}$	3.1k	$119.5_{\pm 0.1}$	$50.6_{\pm 3.3}$	$26.3_{\pm 0.4}$
			SEALS	3.2k	$121.8_{\pm 0.0}$	$64.4_{\pm 0.5}$	$91.5_{\pm 3.4}$	3.1k	$119.8_{\pm 0.0}$	$63.1_{\pm 0.5}$	$89.2_{\pm 3.8}$
		Entropy	AnchorAL	3.7k	$142.4_{\pm 0.0}$	$78.1_{\pm 1.0}$	$4.4_{\pm 0.1}$	3.1k	$119.8_{\pm 0.1}$	$64.9_{\pm 0.9}$	$3.6_{\pm 0.1}$
			No-Op	475	$14.6_{\pm 0.0}$	$2.9_{\pm 0.0}$	6h	475	$14.6_{\pm 0.0}$	$2.9_{\pm 0.0}$	6h
			RandSub	3.6k	$138.3_{\pm 0.1}$	$65.0_{\pm 3.2}$	$29.0_{\pm 0.2}$	3.1k	$119.6_{\pm 0.1}$	$54.3_{\pm 2.0}$	$25.1_{\pm 0.1}$

			SEALS	3.2k	123.7$_{\pm0.0}$	65.6$_{\pm0.8}$	85.4$_{\pm1.3}$	3.1k	119.8$_{\pm0.0}$	63.7$_{\pm1.4}$	81.3$_{\pm2.4}$
		FT-BERTKM	AnchorAL	3.8k	143.3$_{\pm0.0}$	73.8$_{\pm1.5}$	4.5$_{\pm0.1}$	3.1k	119.7$_{\pm0.0}$	59.8$_{\pm1.7}$	3.7$_{\pm0.1}$
			No-Op	425	12.7$_{\pm0.0}$	2.0$_{\pm0.0}$	6h	425	12.7$_{\pm0.0}$	2.0$_{\pm0.0}$	6h
			RandSub	3.6k	138.0$_{\pm0.1}$	52.0$_{\pm8.0}$	31.6$_{\pm0.3}$	3.1k	119.4$_{\pm0.1}$	45.8$_{\pm6.2}$	27.5$_{\pm0.4}$
			SEALS	3.1k	119.6$_{\pm0.1}$	56.5$_{\pm5.6}$	101.6$_{\pm0.9}$	3.1k	119.6$_{\pm0.1}$	56.5$_{\pm5.6}$	101.6$_{\pm0.9}$
		Random	No-Op	3.7k	140.2$_{\pm0.1}$	20.8$_{\pm2.6}$	0.3$_{\pm0.0}$	3.1k	118.8$_{\pm0.1}$	17.0$_{\pm2.4}$	0.2$_{\pm0.0}$
	BERT-tiny	Entropy	AnchorAL	5.0k	191.7$_{\pm0.4}$	63.7$_{\pm21.3}$	2.5$_{\pm0.2}$	3.6k	136.7$_{\pm0.4}$	33.0$_{\pm19.9}$	1.6$_{\pm0.1}$
			No-Op	350	9.7$_{\pm0.0}$	0.0$_{\pm0.0}$	6h	350	9.7$_{\pm0.0}$	0.0$_{\pm0.0}$	6h
			RandSub	4.7k	177.6$_{\pm0.1}$	40.5$_{\pm5.6}$	14.2$_{\pm0.1}$	3.6k	136.6$_{\pm0.1}$	27.5$_{\pm3.8}$	10.9$_{\pm0.1}$
			SEALS	3.6k	136.9$_{\pm0.1}$	48.1$_{\pm8.9}$	39.6$_{\pm1.4}$	3.6k	136.9$_{\pm0.1}$	48.1$_{\pm8.9}$	39.6$_{\pm1.4}$
	DeBERTa-base	Entropy	AnchorAL	1.1k	41.1$_{\pm0.0}$	18.1$_{\pm1.1}$	1.9$_{\pm0.1}$	1.1k	40.2$_{\pm0.0}$	17.6$_{\pm1.2}$	1.9$_{\pm0.1}$
			RandSub	1.1k	40.9$_{\pm0.2}$	5.7$_{\pm12.0}$	14.0$_{\pm0.1}$	1.1k	40.0$_{\pm0.2}$	5.4$_{\pm11.6}$	13.7$_{\pm0.1}$
			SEALS	1.1k	40.2$_{\pm0.0}$	18.0$_{\pm1.5}$	25.7$_{\pm1.4}$	1.1k	40.2$_{\pm0.0}$	18.0$_{\pm1.5}$	25.7$_{\pm1.4}$
	GPT-2	Entropy	AnchorAL	3.6k	136.1$_{\pm0.0}$	0.0$_{\pm0.0}$	3.7$_{\pm0.0}$	3.2k	119.6$_{\pm0.0}$	0.0$_{\pm0.0}$	3.7$_{\pm1.0}$
			RandSub	3.5k	131.3$_{\pm0.0}$	0.0$_{\pm0.0}$	33.2$_{\pm0.1}$	3.2k	119.6$_{\pm0.0}$	0.0$_{\pm0.0}$	30.3$_{\pm0.2}$
			SEALS	3.2k	119.6$_{\pm0.0}$	0.0$_{\pm0.0}$	82.1$_{\pm0.4}$	3.2k	119.6$_{\pm0.0}$	0.0$_{\pm0.0}$	82.1$_{\pm0.4}$
	T5-base	Entropy	AnchorAL	2.3k	86.0$_{\pm0.1}$	24.1$_{\pm2.6}$	4.3$_{\pm0.1}$	2.2k	82.1$_{\pm0.1}$	23.2$_{\pm5.9}$	4.2$_{\pm0.2}$
			RandSub	2.4k	89.5$_{\pm0.0}$	0.0$_{\pm0.0}$	43.9$_{\pm0.2}$	2.2k	81.7$_{\pm0.0}$	0.0$_{\pm0.2}$	40.2$_{\pm0.2}$
			SEALS	2.2k	81.7$_{\pm0.0}$	0.0$_{\pm0.0}$	106.5$_{\pm1.9}$	2.2k	81.7$_{\pm0.0}$	0.0$_{\pm0.0}$	106.5$_{\pm1.9}$
Amazon-Multi	BERT-base	BADGE	AnchorAL	3.6k	110.5$_{\pm1.1}$	82.3$_{\pm14.2}$	8.6$_{\pm0.2}$	3.1k	93.8$_{\pm1.9}$	68.1$_{\pm13.5}$	7.2$_{\pm0.2}$
			No-Op	425	9.6$_{\pm0.2}$	5.0$_{\pm3.0}$	6h	425	9.6$_{\pm0.2}$	5.0$_{\pm3.0}$	6h
			RandSub	3.5k	106.0$_{\pm1.2}$	76.6$_{\pm14.6}$	30.7$_{\pm0.3}$	3.1k	93.0$_{\pm1.0}$	65.5$_{\pm16.1}$	26.9$_{\pm0.1}$
			SEALS	3.1k	93.2$_{\pm0.8}$	66.6$_{\pm14.3}$	97.4$_{\pm2.3}$	3.1k	93.2$_{\pm0.8}$	66.6$_{\pm14.3}$	97.4$_{\pm2.3}$
		Entropy	AnchorAL	3.6k	109.2$_{\pm1.7}$	77.9$_{\pm9.9}$	7.8$_{\pm0.2}$	3.1k	92.7$_{\pm1.7}$	65.3$_{\pm12.4}$	6.7$_{\pm0.4}$
			No-Op	475	10.8$_{\pm0.3}$	4.7$_{\pm3.2}$	6h	475	10.8$_{\pm0.3}$	4.7$_{\pm3.2}$	6h
			RandSub	3.5k	106.2$_{\pm0.6}$	74.7$_{\pm14.2}$	26.6$_{\pm0.3}$	3.1k	92.8$_{\pm0.6}$	65.0$_{\pm12.8}$	23.4$_{\pm0.4}$
			SEALS	3.2k	95.3$_{\pm1.6}$	69.2$_{\pm14.7}$	83.8$_{\pm2.7}$	3.1k	93.2$_{\pm1.6}$	67.8$_{\pm12.6}$	81.6$_{\pm3.6}$
		FT-BERTKM	AnchorAL	3.6k	108.8$_{\pm1.1}$	75.6$_{\pm20.7}$	7.8$_{\pm0.3}$	3.1k	91.2$_{\pm2.8}$	62.7$_{\pm21.3}$	6.4$_{\pm0.3}$
			No-Op	400	8.6$_{\pm0.3}$	3.5$_{\pm3.1}$	6h	400	8.6$_{\pm0.3}$	3.5$_{\pm3.1}$	6h
			RandSub	3.5k	105.0$_{\pm1.1}$	72.3$_{\pm13.2}$	29.1$_{\pm0.6}$	3.1k	91.8$_{\pm0.7}$	61.9$_{\pm11.5}$	25.8$_{\pm0.6}$
			SEALS	3.1k	94.2$_{\pm1.5}$	67.4$_{\pm18.3}$	88.2$_{\pm1.1}$	3.1k	92.6$_{\pm1.5}$	66.3$_{\pm18.1}$	85.8$_{\pm1.5}$
		Random	No-Op	3.6k	104.0$_{\pm3.7}$	63.9$_{\pm37.0}$	0.3$_{\pm0.0}$	3.1k	89.4$_{\pm3.5}$	54.2$_{\pm32.7}$	0.2$_{\pm0.0}$
WikiToxic	BERT-base	BADGE	AnchorAL	4.1k	126.2$_{\pm0.3}$	118.2$_{\pm0.9}$	2.7$_{\pm0.0}$	3.5k	106.6$_{\pm0.7}$	99.4$_{\pm1.5}$	2.2$_{\pm0.0}$
			No-Op	2.9k	85.8$_{\pm0.0}$	77.0$_{\pm0.0}$	6h	2.9k	85.8$_{\pm0.0}$	77.0$_{\pm0.0}$	6h
			RandSub	3.9k	119.5$_{\pm0.4}$	109.0$_{\pm0.9}$	21.8$_{\pm0.1}$	3.5k	104.8$_{\pm0.7}$	94.6$_{\pm1.5}$	19.3$_{\pm0.1}$
			SEALS	3.6k	110.0$_{\pm0.6}$	101.7$_{\pm1.4}$	67.1$_{\pm0.5}$	3.5k	105.9$_{\pm0.6}$	97.7$_{\pm1.4}$	63.6$_{\pm0.5}$
		Entropy	AnchorAL	4.2k	128.1$_{\pm0.6}$	119.0$_{\pm1.2}$	2.6$_{\pm0.0}$	3.5k	106.6$_{\pm0.5}$	99.0$_{\pm1.1}$	2.1$_{\pm0.0}$
			No-Op	2.9k	88.3$_{\pm0.4}$	80.8$_{\pm1.0}$	6h	2.9k	88.3$_{\pm0.4}$	80.8$_{\pm1.0}$	6h

		RandSub	4.0k	$119.4_{\pm 0.2}$	$107.4_{\pm 0.8}$	$20.4_{\pm 0.8}$	3.5k	$104.7_{\pm 0.4}$	$94.6_{\pm 1.4}$	$17.8_{\pm 0.1}$
		SEALS	3.6k	$109.6_{\pm 0.7}$	$100.6_{\pm 1.8}$	$60.9_{\pm 1.5}$	3.5k	$105.9_{\pm 0.5}$	$97.6_{\pm 1.3}$	$57.0_{\pm 1.1}$
	FT-BERTKM	AnchorAL	4.0k	$122.3_{\pm 0.5}$	$113.9_{\pm 1.3}$	$2.6_{\pm 0.0}$	3.5k	$106.8_{\pm 1.1}$	$99.9_{\pm 2.5}$	$2.2_{\pm 0.0}$
		No-Op	2.5k	$73.5_{\pm 0.8}$	$62.0_{\pm 3.0}$	$198.2_{\pm 2.2}$	2.5k	$73.5_{\pm 0.8}$	$62.0_{\pm 3.0}$	$198.2_{\pm 2.2}$
		RandSub	3.8k	$113.4_{\pm 0.3}$	$100.4_{\pm 0.7}$	$22.8_{\pm 0.1}$	3.5k	$103.2_{\pm 0.4}$	$91.3_{\pm 0.8}$	$20.8_{\pm 0.2}$
		SEALS	3.5k	$104.1_{\pm 0.6}$	$92.9_{\pm 2.1}$	$75.7_{\pm 1.1}$	3.5k	$104.1_{\pm 0.6}$	$92.9_{\pm 2.1}$	$75.7_{\pm 1.1}$
	Random	No-Op	4.0k	$108.6_{\pm 0.6}$	$72.4_{\pm 2.1}$	$0.0_{\pm 0.0}$	3.5k	$93.5_{\pm 0.5}$	$60.6_{\pm 2.4}$	$0.0_{\pm 0.0}$