Small-Text: Active Learning for Text Classification in Python

Christopher Schröder[1], **Lydia Müller**[1,2], **Andreas Niekler**[1], **Martin Potthast**[1,3]

[1]Leipzig University
[2]Institute for Applied Informatics (InfAI), Leipzig
[3]ScaDS.AI

Abstract

We introduce `small-text`, an easy-to-use active learning library, which offers pool-based active learning for single- and multi-label text classification in Python. It features numerous pre-implemented state-of-the-art query strategies, including some that leverage the GPU. Standardized interfaces allow the combination of a variety of classifiers, query strategies, and stopping criteria, facilitating a quick mix and match, and enabling a rapid and convenient development of both active learning experiments and applications. With the objective of making various classifiers and query strategies accessible for active learning, `small-text` integrates several well-known machine learning libraries, namely `scikit-learn`, `PyTorch`, and Hugging Face `transformers`. The latter integrations are optionally installable extensions, so GPUs can be used but are not required. Using this new library, we investigate the performance of the recently published SetFit training paradigm, which we compare to vanilla transformer fine-tuning, finding that it matches the latter in classification accuracy while outperforming it in area under the curve. The library is available under the MIT License at https://github.com/webis-de/small-text, in version 1.3.0 at the time of writing.

1 Introduction

Text classification, like most modern machine learning applications, requires large amounts of training data to achieve state-of-the-art effectiveness. However, in many real-world use cases, labeled data does not exist and is expensive to obtain, especially when domain expertise is required. *Active Learning* (Lewis and Gale, 1994) solves this problem by repeatedly selecting unlabeled data instances that are deemed informative according to a so-called *query strategy*, and then having them labeled by an expert (see Figure 1a). A new model is then trained on all previously labeled data, and this process is repeated until a specified stopping criterion is met.

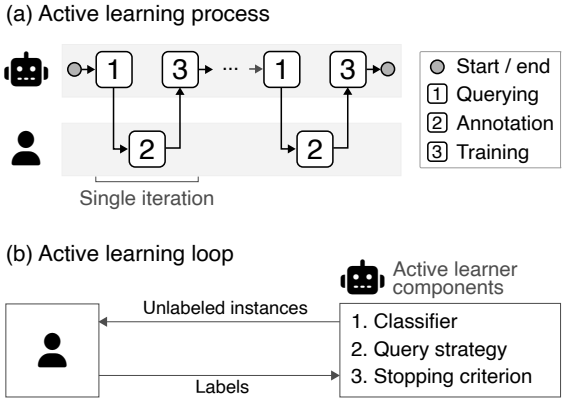

Figure 1: Illustrations of (a) the active learning process, and (b) the active learning setup with the components of the active learner.

Active learning aims to minimize the amount of labeled data required while maximizing the effectiveness (increase per iteration) of the model, e.g., in terms of classification accuracy.

An active learning setup, as shown in Figure 1b, generally consists of up to three components on the system side: a classifier, a query strategy, and an optional stopping criterion. Meanwhile, many approaches for each of these components have been proposed and studied. Determining appropriate combinations of these approaches is only possible experimentally, and efficient implementations are often nontrivial. In addition, the components often depend on each other, for example, when a query strategy relies on parts specific to certain model classes, such as gradients (Ash et al., 2020) or embeddings (Margatina et al., 2021). The more such non-trivial combinations are used together, the more the reproduction effort increases, making a modular library essential.

An obvious solution to the above problems is the use of open source libraries, which, among other benefits, accelerate research and facilitate technology transfer between researchers as well as into practice (Sonnenburg et al., 2007). While solu-

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics
System Demonstrations, pages 84–95

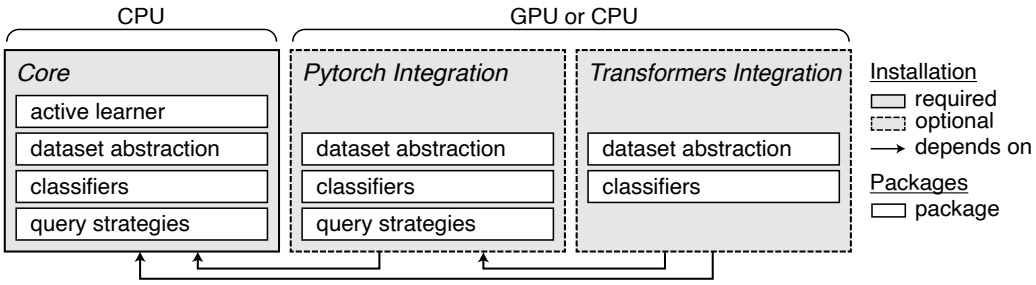

Figure 2: Module architecture of `small-text`. The core module can optionally be extended with a `PyTorch` and `transformers` integration, which enable to use GPU-based models and state-of-the-art transformer-based text classifiers of the Hugging Face `transformers` library, respectively. The dependencies between the module's packages have been omitted.

tions for active learning in general already exist, few address text classification, which requires features specific to natural language processing, such as word embeddings (Mikolov et al., 2013) or language models (Devlin et al., 2019). To fill this gap, we introduce `small-text`, an active learning library that provides tried and tested components for both experiments and applications.

2 Overview of Small-Text

The main goal of `small-text` is to offer state-of-the-art active learning for text classification in a convenient and robust way for both researchers and practitioners. For this purpose, we implemented a modular pool-based active learning mechanism, illustrated in Figure 2, which exposes interfaces for classifiers, query strategies, and stopping criteria. The core of `small-text` integrates `scikit-learn` (Pedregosa et al., 2011), enabling direct use of its classifiers. Overall, the library provides thirteen query strategies, including some that are only usable on text data, five stopping criteria, and two integrations of well-known machine learning libraries, namely `PyTorch` (Paszke et al., 2019) and `transformers` (Wolf et al., 2020). The integrations ease the use of CUDA-based GPU computing and transformer models, respectively. The modular architecture renders both integrations completely optional, resulting in a slim core that can also be used in a CPU-only scenario without unnecessary dependencies. Given the ability to combine a considerable variety of classifiers and query strategies, we can easily build a vast number of combinations of active learning setups.

The library provides relevant text classification baselines such as SVM (Joachims, 1998) and KimCNN (Kim, 2014), and many more can be used through `scikit-learn`. Recent transformer models such as BERT (Devlin et al., 2019) are available through the `transformers` integration. This integration also includes a wrapper that enables the use of the recently published SetFit training paradigm (Tunstall et al., 2022), which uses contrastive learning to fine-tune SBERT embeddings (Reimers and Gurevych, 2019) in a sample efficient manner.

As the query strategy, which selects the instances to be labeled, is the most salient component of an active learning setup, the range of alternative query strategies provided covers four paradigms at the time of writing: (i) confidence-based strategies: least confidence (Lewis and Gale, 1994; Culotta and McCallum, 2005), prediction entropy (Roy and McCallum, 2001), breaking ties (Luo et al., 2005), BALD (Houlsby et al., 2011), CVIRS (Reyes et al., 2018), and contrastive active learning (Margatina et al., 2021); (ii) embedding-based strategies: BADGE (Ash et al., 2020), BERT k-means (Yuan et al., 2020), discriminative active learning (Gissin and Shalev-Shwartz, 2019), and SEALS (Coleman et al., 2022); (iii) gradient-based strategies: expected gradient length (EGL; Settles et al., 2007), EGL-word (Zhang et al., 2017), and EGL-sm (Zhang et al., 2017); and (iv) coreset strategies: greedy coreset (Sener and Savarese, 2018) and lightweight coreset (Bachem et al., 2018). Since there is an abundance of query strategies, this list will likely never be exhaustive—also because strategies from other domains, such as computer vision, are not always applicable to the text domain, e.g., when using the geometry of images (Konyushkova et al., 2015), and thus will be disregarded here.

Furthermore, `small-text` includes a considerable amount of different stopping criteria: (i) stabilizing predictions (Bloodgood and Vijay-Shanker, 2009), (iv) overall-uncertainty (Zhu et al., 2008), (iii) classification-change (Zhu et al., 2008),

(ii) predicted change of F-measure (Altschuler and Bloodgood, 2019), and (v) a criterion that stops after a fixed number of iterations. Stopping criteria are often neglected in active learning although they exert a strong influence on labeling efficiency.

The library is available via the python packaging index and can be installed with just a single command: `pip install small-text`. Similarly, the integrations can be enabled using the extra requirements argument of Python's `setuptools`, e.g., the `transformers` integration is installed using `pip install small-text[transformers]`. The robustness of the implementation rests on extensive unit and integration tests. Detailed examples, an API documentation, and common usage patterns are available in the online documentation.[1]

3 Library versus Annotation Tool

We designed `small-text` for two types of settings: (i) experiments, which usually consist of either automated active learning evaluations or short-lived setups with one or more human annotators, and (ii) real-world applications, in which the final model is subsequently applied on unlabeled or unseen data. Both cases benefit from a library which offers a wide range of well-tested functionality.

To clarify on the distinction between a library and an annotation tool, `small-text` is a library, by which we mean a reusable set of functions and classes that can be used and combined within more complex programs. In contrast, annotation tools provide a graphical user interface and focus on the interaction between the user and the system. Obviously, `small-text` is still intended to be used by annotation tools but remains a standalone library. In this way it can be used (i) in combination with an annotation tool, (ii) within an experiment setting, or (iii) as part of a backend application, e.g. a web API. As a library it remains compatible to all of these use cases. This flexibility is supported by the library's modular architecture which is also in concordance with software engineering best practices, where high cohesion and low coupling (Myers, 1975) are known to contribute towards highly reusable software (Müller et al., 1993; Tonella, 2001). As a result, `small-text` should be compatible with most annotations tools that are extensible and support text classification.

[1] https://small-text.readthedocs.io

4 Code Example

In this section we show a code example to perform active learning with transformers models.

Dataset First, we create (for the sake of a simple example) a synthetic two-class spam dataset of 100 instances. The data is given by a list of texts and a list of integer labels. To define the tokenization strategy, we provide a `transformers` tokenizer. From these individual parts we construct a `TransformersDataset` object which is a dataset abstraction that can be used by the interfaces in `small-text`. This yields a binary text classification dataset containing 50 examples of the positive class (spam) and the negative class (ham) each:

```
import numpy as np
from small_text import TransformersDataset, \
    TransformerModelArguments
from transformers import AutoTokenizer

# Fake data example:
#   50 spam and 50 non-spam examples
text = np.array(['this is ham'] * 50 +
                ['this is spam'] * 50)
labels = np.array([0] * 50 + [1] * 50)

transformer_model = 'bert-base-uncased'
tokenizer = AutoTokenizer.from_pretrained(
    transformer_model)

train = TransformersDataset.from_arrays(
    text, labels, tokenizer,
    target_labels=np.array([0, 1]),
    max_length=10
)
```

Active Learning Configuration Next, we configure the classifier and query strategy. Although the active learner, query strategies, and stopping criteria components are dataset- and classifier-agnostic, classifier and dataset have to match (i.e. `TransformerBasedClassification` must be used with `TransformersDataset`) owing to the different underlying data structures:

```
from small_text import LeastConfidence, \
    TransformerBasedClassificationFactory \
    as TransformerFactory
num_classes = 2
model_args = TransformerModelArguments(
    transformer_model)

clf_factory = TransformerFactory(model_args,
    num_classes, kwargs={'device': 'cuda'})
query_strategy = LeastConfidence()
```

Since the active learner may need to instantiate a new classifier before the training step, a factory (Gamma et al., 1995) is responsible for creating new classifiers. Finally, we set the query strategy to least confidence (Culotta and McCallum, 2005).

Initialization There is a chicken-and-egg problem for active learning because most query strategies rely on the model, and a model in turn is trained on labeled instances which are selected by the query strategy. This problem can be solved by either providing an initial model (e.g. through manual labeling), or by using cold start approaches (Yuan et al., 2020). In this example we simulate a user-provided initialization by looking up the respective true labels and providing an initial model:

```
from small_text import \
    PoolBasedActiveLearner, \
    random_initialization_balanced as init

active_learner = PoolBasedActiveLearner(
    clf_factory, query_strategy, train)

# Provide initial data.
indices_initial = init(train.y, n_samples=10)

active_learner.initialize_data(
    indices_initial,
    train.y[indices_initial]
)
```

To provide an initial model in the experimental scenario (where true labels are accessible), `small-text` provides sampling methods, from which we use the balanced sampling to obtain a subset whose class distribution is balanced (or close thereto). In a real-world application, initialization would be accomplished through a starting set of labels supplied by the user. Alternatively, a cold start classifier or query strategy can be used instead.

Active Learning Loop After the previous code examples prepared the setting by loading a dataset, configuring the active learning setup, and providing an initial model, the following code block shows the actual active learning loop. In this example, we perform five queries during each of which ten instances are queried. During a query step the query strategy samples instances to be labeled. Subsequently, new labels for each instance are provided and passed to the `update` method, and then a new model is trained. In this example, it is a simulated response relying on true labels, but in a real-world application this part is the user's response.

```
from sklearn.metrics import accuracy_score

num_queries = 5
for i in range(num_queries):
    # Query 10 samples per iteration.
    indices_queried = active_learner.query(
        num_samples=10
    )

    # Simulate user interaction here.
    #   Replace this for real-world usage.
    y = train.y[indices_queried]

    # Provide labels for the queried indices.
    active_learner.update(y)

    # Evaluate accuracy on the train set
    print(f'Iteration {i+1}')
    y_pred = active_learner.classifier\
        .predict(train)
    print('Train accuracy: {:.2f}'.format(
        accuracy_score(y_pred, train.y)))
```

In summary, we built a full active learning setup in only very few lines of code. The actual active learning loop consists of just the previous code block and changing hyperparameters, e.g., using a different query strategy, is as easy as adapting the `query_strategy` variable.

5 Comparison to Previous Software

Unsurprisingly, after decades of research and development on active learning, numerous other libraries are available that focus on active learning as well. In the following we present a selection of the most relevant open-source projects for which either a related publication is available or a larger user base exists: `JCLAL` (Reyes et al., 2016) is a generic framework for active learning which is implemented in Java and can be used either through XML configurations or directly from the code. It offers an experimental setting which includes 18 query strategies. The aim of `libact` (Yang et al., 2017) is to provide active learning for real-world applications. Among 19 other strategies, it includes a well-known meta-learning strategy (Hsu and Lin, 2015). `BaaL` (Atighehchian et al., 2020) provides bayesian active learning including methods to obtain uncertainty estimates. The `modAL` library (Danka and Horvath, 2018) offers single- and multi-label active learning, provides 12 query strategies, also builds on `scikit-learn` by default, and offers instructions how to include GPU-based models using `Keras` and `PyTorch`. `ALiPy` (Tang et al., 2019) provides an active learning framework targeted at

Name	Active Learning			Code					
	QS	SC	Text Focus	GPU support	Unit Tests	Language	License	Last Update	Repository
JCLAL[1]	18	2	✗	✗	✗	Java	GPL	2017	
libact[2]	19	-	✗	✗	✓	Python	BSD-2-Clause	2021	
modAL[3]	12	-	✗	✓	✓	Python	MIT	2022	
ALiPy[4]	22	4	✗	✗	✓	Python	BSD-3-Clause	2022	
BaaL[5]	9	-	✗	✓	✓	Python	Apache 2.0	2023	
lrtc[6]	7	-	✓	✓	✗	Python	Apache 2.0	2021	
scikit-activeml[7]	29	-	✗	✓	✓	Python	BSD-3-Clause	2023	
ALToolbox[8]	19	-	✓	✓	✓	Python	MIT	2023	
small-text	14	5	✓	✓	✓	Python	MIT	2023	

Table 1: Comparison between small-text and relevant previous active learning libraries. We abbreviated the number of query strategies by "QS", the number of stopping criteria by "SC", and the low-resource-text-classification framework by lrtc. All information except "Publication Year" and "Code Repository" has been extracted from the linked GitHub repository of the respective library on February 24th, 2023. Random baselines were not counted towards the number of query strategies. Publications: [1]Reyes et al. (2016), [2]Yang et al. (2017), [3]Danka and Horvath (2018), [4]Tang et al. (2019), [5]Atighehchian et al. (2020), [6]Ein-Dor et al. (2020), [7]Kottke et al. (2021), [8]Tsvigun et al. (2022).

the experimental active learning setting. Apart from providing 22 query strategies, it supports alternative active learning settings, e.g., active learning with noisy annotators. The low-resource-text-classification-framework (lrtc; (Ein-Dor et al., 2020)) is an experimentation framework for the low resource scenario and supports which can be easily extended. It also focuses on text classification and has a number of built-in models, datasets, and query strategies to perform active learning experiments. Another recent library is scikit-activeml which offers general active learning built around scikit-learn. It comes with 29 query strategies but provides no stopping criteria. GPU-based functionality can be used via skorch,[2] a PyTorch wrapper, which is a ready-to-use adapter as opposed to our implemented classifier structures but is on the other hand restricted to the scikit-learn interfaces. ALToolbox (Tsvigun et al., 2022) is an active learning framework that provides an annotation interface and a benchmarking mechanism to develop new query strategies. While it has some overlap with small-text, it is not a library, but also focuses on text data, namely on text classification and sequence tagging.

In Table 1, we compare small-text to the previously mentioned libraries, and compare them based on several criteria related to active learning or to the respective code base: While all libraries provide a selection of query strategies, not all libraries offer stopping criteria, which are crucial to reducing the total annotation effort and thus directly influence the efficiency of the active learning process (Vlachos, 2008; Laws and Schütze, 2008; Olsson and Tomanek, 2009). We can also see a difference in the number of provided query strategies. While a higher number of query strategies is certainly not a disadvantage, it is more important to provide the most relevant strategies (either due to recency, domain-specificity, strong general performance, or because it is a baseline). Based on these criteria, small-text provides numerous recent strategies such as BADGE (Ash et al., 2020), BERT K-Means (Yuan et al., 2020), and contrastive active learning (Margatina et al., 2021), as well as the gradient-based strategies by Zhang et al. (2017), where the latter are unique to active learning for text classification. Selecting a subset of query strategies is especially important since active learning experiments are computationally expensive (Margatina et al., 2021; Schröder et al., 2022), and therefore not every strategy can be tested in the context of an experiment or application. Finally, only small-text, lrtc, and ALToolbox focus on text, and only about half of the libraries offer access to GPU-based deep learning, which has become indispensable for text classification due to the recent advances and ubiquity of transformer-based models (Vaswani et al., 2017; Devlin et al., 2019).

The distinguishing characteristic of small-text is the focus on text classification, paired with a multitude of interchangeable components. It of-

[2]We also evaluated the use of skorch but transformer models were not supported at that time.

Dataset Name (ID)	Type	Classes	Training	Test
AG's News[1] (AGN)	N	4	120,000	⋆7,600
Customer Reviews[2] (CR)	S	2	3,397	378
Movie Reviews[3] (MR)	S	2	9,596	1,066
Subjectivity[4] (SUBJ)	S	2	9,000	1,000
TREC-6[5] (TREC-6)	Q	6	5,500	⋆500

Table 2: Key characteristics about the examined datasets: [1]Zhang et al. (2015), [2]Hu and Liu (2004), [3]Pang and Lee (2005), [4]Pang and Lee (2004), [5]Li and Roth (2002). The dataset type was abbreviated by N (News), S (Sentiment), Q (Questions). ⋆: Predefined test sets were available and adopted.

fers the most comprehensive set of features (as shown in Table 1) and through the integrations these components can be mixed and matched to easily build numerous different active learning setups, with or without leveraging the GPU. Finally, it allows to use concepts from natural language processing (such as transformer models) and provides query strategies unique to text classification.

6 Experiment

We perform an active learning experiment comparing an SBERT model trained with the recent sentence transformers fine-tuning paradigm (SetFit; (Tunstall et al., 2022)) over a BERT model trained with standard fine-tuning. SetFit is a contrastive learning approach that trains on pairs of (dis)similar instances. Given a fixed amount of differently labeled instances, the number of possible pairs is considerably higher than the size of the original set, making this approach highly sample efficient (Chuang et al., 2020; Hénaff, 2020) and therefore interesting for active learning.

Setup We reproduce the setup of our previous work (Schröder et al., 2022) and evaluate on the datasets shown in Table 2 with an extended set of query strategies. Starting with a pool-based active learning setup with 25 initial samples, we perform 20 queries during each of which 25 instances are queried and labeled. Since SetFit has only been evaluated for single-label classification (Tunstall et al., 2022), we focus on single-label classification as well. The goal is to compare the following two models: (i) BERT (`bert-large-uncased`; (Devlin et al., 2019)) with 336M parameters and (ii) SBERT (`paraphrase-mpnet-base-v2`; (Reimers and Gurevych, 2019)) with 110M parameters. The first model is trained via vanilla fine-tuning and the second using SetFit. For the sake of brevity, we refer to the first as "BERT" and to the second as "SetFit". To compare their performance during active learning, we provide an extensive benchmark over multiple computationally inexpensive uncertainty-based query strategies, which were selected due to encouraging results in our previous work. Moreover, we include BALD, BADGE, and greedy coreset—all of which are computationally more expensive, but have been increasingly used in recent work (Ein-Dor et al., 2020; Yu et al., 2022).

Model	Strategy	Rank		Result	
		Acc.	AUC	Acc.	AUC
BERT	PE	2.20	2.80	0.917	0.858
	BT	**1.40**	**1.60**	**0.919**	**0.868**
	LC	3.80	3.20	0.916	0.863
	CA	4.20	5.00	0.915	0.857
	BA	3.00	5.20	0.917	0.855
	BD	6.20	4.60	0.909	0.862
	CS	6.60	7.60	0.910	0.843
	RS	7.80	5.40	0.901	0.856
SetFit	PE	2.80	3.20	0.927	0.906
	BT	2.80	**1.60**	0.926	**0.912**
	LC	**2.20**	2.60	**0.927**	0.908
	CA	4.80	3.80	0.924	0.907
	BA	5.20	6.20	0.923	0.902
	BD	6.60	5.60	0.915	0.904
	CS	2.80	4.40	0.927	0.909
	RS	6.60	6.80	0.907	0.899

Table 3: The "Rank" columns show the mean rank when ordered by mean accuracy (Acc.) and by area under curve (AUC). The "Result" columns show the mean accuracy and AUC. All values used in this table refer to state after the final iteration. Query strategies are abbreviated as follows: prediction entropy (PE), breaking ties (BT), least confidence (LC), contrastive active learning (CA), BALD (BA), BADGE (BD), greedy coreset (CS), and random sampling (RS).

Results In Table 3, the results show the summarized classification performance in terms of (i) final accuracy after the last iteration, and (ii) area under curve (AUC). We also compare strategies by ranking them from 1 (best) to 8 (worst) per model and dataset by accuracy and AUC. First, we can also confirm for SetFit the earlier finding that uncertainty-based strategies perform strong for BERT (Schröder et al., 2022). Second, SetFit configurations result in between 0.06 and 1.7 percentage points higher mean accuracy, and also in betwen 4.2 and 6.6 higher AUC when averaged over model and query strategy. Interestingly, the greedy coreset strategy (CS) is remarkably more successful for the SetFit runs compared to the BERT runs. Detailed results per configuration can be found

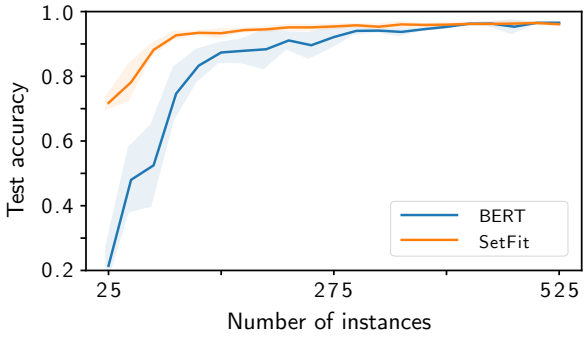

Figure 3: An exemplary learning curve showing the difference in test accuracy for breaking ties strategy on the TREC dataset, comparing BERT and SetFit. The tubes represent the standard deviation across five runs.

in the appendix, where it can be seen that SetFit reaches higher accuracy scores in most configurations, and better AUC scores in all configurations.

Discussion When trained with the new SetFit paradigm, models having only a third of the parameters compared to the large BERT model achieve results that are not only competitive, but slightly better regarding final accuracy and considerably better in terms of AUC. Since the final accuracy values are often within one percentage point or less to each other, it is obvious that the improvement in AUC stems from improvements in earlier queries, i.e. steeper learning curves. We suspect that this is at least partly owed to sample efficiency from SetFit's training that uses pairs of instances. Moreover, this has the additional benefit of reducing instability of transformer models (Mosbach et al., 2021) as can be exemplarily seen in Figure 3. This increasingly occurs when the training set is small (Mosbach et al., 2021), which is likely alleviated with the additional instance pairs. On the other hand, training cost increase linearly with the number of pairs per instance. In the low-data regime, however, this is a manageable additional cost that is worth the benefits.

7 Library Adoption

As recent publications have already adopted `small-text`, we present four examples which have already successfully utilized it for their experiments.

Abusive Language Detection Kirk et al. (2022) investigated the detection of abusive language using transformer-based active learning on six datasets of which two exhibited a balanced and four an imbalanced class distribution. They evaluated a pool-based binary active learning setup, and their main finding is that, when using active learning, a model for abusive language detection can be efficiently trained using only a fraction of the data.

Classification of Citizens' Contributions In order to support the automated classification of German texts from online citizen participation processes, Romberg and Escher (2022) used active learning to classify texts collected by three cities into eight different topics. They evaluated this real-world dataset both as a single- and multi-label active learning setup, finding that active learning can considerably reduce the annotation efforts.

Softmax Confidence Estimates Gonsior et al. (2022) examined several alternatives to the softmax function to obtain better confidence estimates for active learning. Their setup extended small-text to incorporate additional softmax alternatives and found that confidence-based methods mostly selected outliers. As a remedy to this they proposed and evaluated uncertainty clipping.

Revisiting Uncertainty-Based Strategies In a previous publication, we reevaluated traditional uncertainty-based query strategies with recent transformer models (Schröder et al., 2022). We found that uncertainty-based methods can still be highly effective and that the breaking ties strategy is a drop-in replacement for prediction entropy.

Not only have all of these works successfully applied small-text to a variety of different problems, but each work is also accompanied by a GitHub repository containing the experiment code, which is the outcome we had hoped for. We expect that `small-text` will continue to gain adoption within the active learning and text classification communities, so that future experiments will increasingly rely on it by both reusing existing components and by creating their own extensions, thereby supporting the field through open reproducible research.

8 Conclusion

We introduced `small-text`, a modular Python library, which offers state-of-the-art active learning for text classification. It integrates `scikit-learn`, `PyTorch`, and `transformers`, and provides robust components that can be mixed and matched to quickly apply active learning in both experiments and applications, thereby making active learning easily accessible to the Python ecosystem.

Limitations

Although a library can, among other things, lower the barrier of entry, save time, and speed up research, this can only be leveraged with basic knowledge of the Python programming language. All included algorithmic components are subject to their own limitations, e.g., the greedy coreset strategy quickly becomes computationally expensive as the amount labeled data increases. Moreover, some components have hyperparameters which require an understanding of the algorithm to achieve the best classification performance. In the end, we provide a powerful set of tools which still has to be properly used to achieve the best results.

As `small-text` covers numerous text classification models, query strategies, and stopping criteria, some limitations from natural language processing, text classification and active learning apply as well. For example, all included classification models rely on tokenization, which is inherently more difficult for languages which have no clear word boundaries such as Chinese, Japanese, Korean, or Thai.

Ethics Statement

In this paper, we presented `small-text`, a library which can—like any other software—be used for good or bad. It can be used to bootstrap classification models in scenarios where no labeled data is available. This could be used for good, e.g. for spam detection, hatespeech detection, or targeted news filtering, but also for bad, e.g., for creating models that detect certain topics that are to be censored in authoritarian regimes. While such systems already exist and are of sophisticated quality, `small-text` is unlikely to change anything at this point. On the contrary, being open-source software, these methods can now be used by a larger audience, which contributes towards the democratization of classification algorithms.

Acknowledgments

We thank the anonymous reviewers for their constructive advice and the early adopters of the library for their invaluable feedback.

This research was partially funded by the Development Bank of Saxony (SAB) under project numbers 100335729 and 100400221. Computations were done (in part) using resources of the Leipzig University Computing Centre.

References

Michael Altschuler and Michael Bloodgood. 2019. Stopping active learning based on predicted change of F measure for text classification. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 47–54. IEEE.

Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2020. Deep batch active learning by diverse, uncertain gradient lower bounds. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*. OpenReview.net.

Parmida Atighehchian, Frédéric Branchaud-Charron, and Alexandre Lacoste. 2020. Bayesian active learning for production, a systematic study and a reusable library. *arXiv preprint arXiv:2006.09916*.

Olivier Bachem, Mario Lucic, and Andreas Krause. 2018. Scalable k-Means Clustering via Lightweight Coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (KDD)*, pages 1119–1127.

Michael Bloodgood and K. Vijay-Shanker. 2009. A method for stopping active learning based on stabilizing predictions and the need for user-adjustable stopping. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 39–47, Boulder, Colorado. Association for Computational Linguistics.

Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. 2020. Debiased contrastive learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, volume 33, pages 8765–8775. 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. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 36(6):6402–6410.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*, volume 2, pages 746–751.

Tivadar Danka and Peter Horvath. 2018. modAL: A modular active learning framework for Python. *arXiv preprint arXiv:1805.00979*.

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 American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4171–4186. Association for Computational Linguistics.

Liat Ein-Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active Learning for BERT: An Empirical Study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962, Online. Association for Computational Linguistics.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*, 1 edition. Addison-Wesley Longman Publishing Co., Inc., USA. 37. Reprint.

Daniel Gissin and Shai Shalev-Shwartz. 2019. Discriminative active learning. *arXiv preprint arXiv:1907.06347*.

Julius Gonsior, Christian Falkenberg, Silvio Magino, Anja Reusch, Maik Thiele, and Wolfgang Lehner. 2022. To softmax, or not to softmax: that is the question when applying active learning for transformer models. *arXiv preprint arXiv:2210.03005*.

Olivier J. Hénaff. 2020. Data-efficient image recognition with contrastive predictive coding. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4182–4192. PMLR.

Neil Houlsby, Ferenc Huszar, Zoubin Ghahramani, and Máté Lengyel. 2011. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*.

Wei-Ning Hsu and Hsuan-Tien Lin. 2015. Active learning by learning. *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, 29(1).

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. Association for Computing Machinery.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98, 10th European Conference on Machine Learning, Chemnitz, Germany, April 21-23, 1998, Proceedings*, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142. Springer.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Hannah Kirk, Bertie Vidgen, and Scott Hale. 2022. Is More Data Better? Re-thinking the Importance of Efficiency in Abusive Language Detection with Transformers-Based Active Learning. In *Proceedings of the Third Workshop on Threat, Aggression and Cyberbullying (TRAC 2022)*, pages 52–61, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. 2015. Introducing geometry in active learning for image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2974–2982.

Daniel Kottke, Marek Herde, Tuan Pham Minh, Alexander Benz, Pascal Mergard, Atal Roghman, Christoph Sandrock, and Bernhard Sick. 2021. scikitactiveml: A Library and Toolbox for Active Learning Algorithms. *Preprints.org*.

Florian Laws and Hinrich Schütze. 2008. Stopping criteria for active learning of named entity recognition. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 465–472.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, volume 1 of *COLING '02*, pages 1–7, USA. Association for Computational Linguistics.

Tong Luo, Kurt Kramer, Dmitry B. Goldgof, Lawrence O. Hall, Scott Samson, Andrew Remsen, and Thomas Hopkins. 2005. Active Learning to Recognize Multiple Types of Plankton. *Journal of Machine Learning Research (JMLR)*, 6:589–613.

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 (EMNLP)*, pages 650–663.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations (ICLR)*.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning BERT: misconceptions, explanations, and strong baselines. In *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*. OpenReview.net.

Hausi A. Müller, Mehmet A. Orgun, Scott R. Tilley, and James S. Uhl. 1993. A reverse-engineering approach to subsystem structure identification. *Journal of Software Maintenance: Research and Practice*, 5(4):181–204.

Glenford J. Myers. 1975. *Reliable Software through Composite Design*. Petrocelli/Charter.

Fredrik Olsson and Katrin Tomanek. 2009. An intrinsic stopping criterion for committee-based active learning. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL)*, pages 138–146.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 271–278, Barcelona, Spain.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

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 32*, pages 8024–8035.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research (JMLR)*, 12(85):2825–2830.

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.

Oscar Reyes, Carlos Morell, and Sebastián Ventura. 2018. Effective active learning strategy for multi-label learning. *Neurocomputing*, 273:494–508.

Oscar Reyes, Eduardo Pérez, María del Carmen Rodríguez-Hernández, Habib M. Fardoun, and Sebastián Ventura. 2016. JCLAL: A Java Framework for Active Learning. *Journal of Machine Learning Research (JMLR)*, 17(95):1–5.

Julia Romberg and Tobias Escher. 2022. Automated topic categorisation of citizens' contributions: Reducing manual labelling efforts through active learning. In *Electronic Government*, pages 369–385, Cham. Springer International Publishing.

Nicholas Roy and Andrew McCallum. 2001. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 441–448.

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 (Findings of ACL 2022)*, pages 2194–2203.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Burr Settles, Mark Craven, and Soumya Ray. 2007. Multiple-instance active learning. In *Proceedings of the 20th International Conference on Neural Information Processing Systems (NIPS)*, pages 1289–1296.

Sören Sonnenburg, Mikio L. Braun, Cheng Soon Ong, Samy Bengio, Leon Bottou, Geoffrey Holmes, Yann LeCun, Klaus-Robert Müller, Fernando Pereira, Carl Edward Rasmussen, Gunnar Rätsch, Bernhard Schölkopf, Alexander Smola, Pascal Vincent, Jason Weston, and Robert Williamson. 2007. The Need for Open Source Software in Machine Learning. *Journal of Machine Learning Research (JMLR)*, 8(81):2443–2466.

Ying-Peng Tang, Guo-Xiang Li, and Sheng-Jun Huang. 2019. ALiPy: Active learning in python. *arXiv preprint arXiv:1901.03802*.

Paolo Tonella. 2001. Concept analysis for module restructuring. *IEEE Trans. Software Eng.*, 27(4):351–363.

Akim Tsvigun, Leonid Sanochkin, Daniil Larionov, Gleb Kuzmin, Artem Vazhentsev, Ivan Lazichny, Nikita Khromov, Danil Kireev, Aleksandr Rubashevskii, and Olga Shahmatova. 2022. ALToolbox: A set of tools for active learning annotation of natural language texts. In *Proceedings of the The 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 406–434, Abu Dhabi, UAE. Association for Computational Linguistics.

Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. 2022. Efficient few-shot learning without prompts. *arXiv preprint arXiv:2209.11055*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 5998–6008.

Andreas Vlachos. 2008. A stopping criterion for active learning. *Computer Speech & Language*, 22(3):295–312.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. 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 (EMNLP)*, pages 38–45.

Yao-Yuan Yang, Shao-Chuan Lee, Yu-An Chung, Tung-En Wu, Si-An Chen, and Hsuan-Tien Lin. 2017. libact: Pool-based active learning in python. *arXiv preprint arXiv:1710.00379*.

Yue Yu, Lingkai Kong, Jieyu Zhang, Rongzhi Zhang, and Chao Zhang. 2022. AcTune: Uncertainty-based active self-training for active fine-tuning of pretrained language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1422–1436, Seattle, United States. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Proceedings of the Advances in Neural Information Processing Systems 28 (NIPS)*, pages 649–657. Curran Associates, Inc., Montreal, Quebec, Canada.

Ye Zhang, Matthew Lease, and Byron C. Wallace. 2017. Active discriminative text representation learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pages 3386–3392.

Jingbo Zhu, Huizhen Wang, and Eduard Hovy. 2008. Multi-criteria-based strategy to stop active learning for data annotation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1129–1136, Manchester, UK. Coling 2008 Organizing Committee.

Supplementary Material

A Technical Environment

All experiments were conducted within a Python 3.8 environment. The system had CUDA 11.1 installed and was equipped with an NVIDIA GeForce RTX 2080 Ti (11GB VRAM).

B Experiments

Each experiment configuration represents a combination of model, dataset and query strategy, and has been run for five times.

B.1 Datasets

We used datasets that are well-known benchmarks in text classification and active learning. All datasets are accessible to the Python ecosystem via Python libraries that provide fast access to those datasets. We obtained CR and SUBJ using gluonnlp, and AGN, MR, and TREC using huggingface datasets.

B.2 Pre-Trained Models

In the experiments, we fine-tuned (i) a large BERT model (bert-large-uncased) and (ii) an SBERT paraphrase-mpnet-base model (sentence-transformers/paraphrase-mpnet-base-v2). Both are available via the huggingface model repository.

B.3 Hyperparameters

Maximum Sequence Length We set the maximum sequence length to the minimum multiple of ten, so that 95% of the given dataset's sentences contain at most that many tokens.

Transformer Models For BERT, we adopt the hyperparameters from Schröder et al. (2022). For SetFit, we use the same learning rate and optimizer parameters but we train for only one epoch.

C Evaluation

In Table 4 and Table 5 we report final accuracy and AUC scores including standard deviations, measured after the last iteration. Note that results obtained through PE, BT, and LC are equivalent for binary datasets.

C.1 Evaluation Metrics

Active learning was evaluated using standard metrics, namely accuracy und area under the learning curve. For both metrics, the respective scikit-learn implementation was used.

Dataset	Model	Query Strategy							
		PE	BT	LC	CA	BA	BD	CS	RS
AGN	BERT	0.898 0.003	0.901 0.004	0.900 0.001	0.889 0.010	0.889 0.008	0.894 0.003	0.881 0.006	0.886 0.004
	SetFit	0.900 0.002	0.902 0.004	**0.902 0.002**	0.892 0.006	0.887 0.010	0.896 0.003	0.896 0.003	0.877 0.005
CR	BERT	0.920 0.009	0.920 0.009	0.916 0.006	0.917 0.010	0.919 0.010	0.911 0.010	0.915 0.012	0.902 0.014
	SetFit	0.937 0.014	0.937 0.014	0.937 0.014	0.938 0.009	0.934 0.004	0.913 0.011	**0.939 0.011**	0.912 0.010
MR	BERT	0.850 0.005	0.850 0.005	0.846 0.008	0.844 0.008	0.859 0.003	0.835 0.017	0.843 0.006	0.831 0.020
	SetFit	0.871 0.009	0.871 0.009	0.871 0.009	0.869 0.004	0.867 0.005	0.864 0.008	0.870 0.008	**0.871 0.003**
SUBJ	BERT	0.959 0.005	0.959 0.005	0.958 0.003	0.958 0.008	0.959 0.003	0.948 0.006	0.957 0.004	0.937 0.006
	SetFit	0.962 0.004	0.962 0.004	0.962 0.004	0.960 0.002	**0.966 0.002**	0.942 0.002	0.963 0.003	0.932 0.005
TREC-6	BERT	0.960 0.002	0.966 0.003	0.960 0.008	0.965 0.006	0.958 0.007	0.958 0.009	0.952 0.015	0.947 0.009
	SetFit	0.966 0.005	0.961 0.005	0.966 0.005	0.963 0.008	0.961 0.005	0.958 0.005	**0.967 0.004**	0.946 0.009

Table 4: Final accuracy per dataset, model, and query strategy. We report the mean and standard deviation over five runs. The best result per dataset is printed in bold. Query strategies are abbreviated as follows: prediction entropy (PE), breaking ties (BT), least confidence (LC), contrastive active learning (CA), BALD (BA), BADGE (BD), greedy coreset (CS), and random sampling (RS). The best result per dataset is printed in bold.

Dataset	Model	Query Strategy							
		PE	BT	LC	CA	BA	BD	CS	RS
AGN	BERT	0.827 0.009	0.839 0.014	0.836 0.009	0.821 0.015	0.819 0.012	0.840 0.003	0.804 0.012	0.825 0.011
	SetFit	0.881 0.002	**0.889 0.003**	0.885 0.005	0.879 0.004	0.869 0.006	0.881 0.002	0.881 0.003	0.867 0.004
CR	BERT	0.885 0.007	0.885 0.007	0.881 0.007	0.881 0.011	0.882 0.006	0.876 0.005	0.874 0.011	0.877 0.011
	SetFit	0.925 0.001	0.925 0.001	0.925 0.001	0.927 0.003	0.924 0.005	0.910 0.005	**0.930 0.002**	0.908 0.008
MR	BERT	0.819 0.010	0.819 0.010	0.820 0.007	0.813 0.009	0.817 0.013	0.808 0.011	0.804 0.010	0.813 0.004
	SetFit	0.859 0.004	0.859 0.004	0.859 0.004	**0.859 0.003**	0.858 0.004	0.855 0.002	0.858 0.004	0.857 0.002
SUBJ	BERT	0.944 0.008	0.944 0.008	0.943 0.007	0.940 0.009	0.939 0.009	0.929 0.005	0.934 0.006	0.924 0.007
	SetFit	**0.953 0.002**	**0.953 0.002**	**0.953 0.002**	0.952 0.003	0.950 0.002	0.940 0.003	0.949 0.001	0.935 0.002
TREC-6	BERT	0.818 0.033	0.855 0.023	0.837 0.034	0.829 0.030	0.816 0.029	0.856 0.024	0.799 0.037	0.843 0.008
	SetFit	0.910 0.008	**0.934 0.005**	0.919 0.008	0.917 0.013	0.907 0.017	0.934 0.010	0.927 0.008	0.927 0.004

Table 5: Final area under curve (AUC) per dataset, model, and query strategy. We report the mean and standard deviation over five runs. The best result per dataset is printed in bold. Query strategies are abbreviated as follows: prediction entropy (PE), breaking ties (BT), least confidence (LC), contrastive active learning (CA), BALD (BA), BADGE (BD), greedy coreset (CS), and random sampling (RS). The best result per dataset is printed in bold.

D Library Adoption

As mentioned in Section 7, the experiment code of previous works documents how `small-text` was used and can be found at the following locations:

- Abusive Language Detection: https://github.com/HannahKirk/ActiveTransformers-for-AbusiveLanguage
- Classification of Citizens' Contributions: https://github.com/juliaromberg/egov-2022
- Softmax Confidence Estimates: https://github.com/jgonsior/btw-softmax-clipping
- Revisiting Uncertainty-Based Strategies: https://github.com/webis-de/ACL-22