ALToolbox: A Set of Tools for Active Learning Annotation of Natural Language Texts

Akim Tsvigun[1,3] ♢, Leonid Sanochkin[1,3] ♢, Daniil Larionov[2,5], Gleb Kuzmin[1,5], Artem Vazhentsev[1,4], Ivan Lazichny[1], Nikita Khromov[4], Danil Kireev[5], Aleksandr Rubashevskii[4], Olga O. Shahmatova[7], Dmitry V. Dylov[4], Igor Galitskiy[1], Artem Shelmanov[1,6]

[1]AIRI, [2]MIPT, [3]HSE, [4]Skoltech, [5]FRC CSC RAS, [6]MBZUAI
[7]National Medical Research Centre of Cardiology
{tsvigun, sanochkin, shelmanov}@airi.net
artem.shelmanov@mbzuai.ac.ae

Abstract

We present ALToolbox – an open-source framework for active learning (AL) annotation in natural language processing. Currently, the framework supports text classification, sequence tagging, and seq2seq tasks. Besides state-of-the-art query strategies, ALToolbox provides a set of tools that help to reduce computational overhead and duration of AL iterations and increase annotated data reusability. The framework aims to support data scientists and researchers by providing an easy-to-deploy GUI annotation tool directly in the Jupyter IDE and an extensible benchmark for novel AL methods. We prepare a small demonstration of ALToolbox capabilities available online[1,2]. The code of the framework is published under the MIT license[3].

1 Introduction

The development of text processing applications based on machine learning (ML) usually requires a lot of labeled data. Despite numerous annotated corpora designed for various tasks and available for resource-rich languages, in practice, the business logic of an application is often very specific and cannot be implemented using only public resources. Manual annotation of natural language corpora is a tedious and time-consuming task, which can take up to 30-40% of the application development time.

For rather simple tasks, the annotation of corpora can be organized using crowd-sourcing. However, crowd-sourcing is not suitable for specific domains like medicine, finance, information technology, or any other field that requires specific qualifications or knowledge of business logic. It is also problematic to apply crowd-sourcing when the annotation scheme is complex and requires some premature training of annotators. In each of the aforementioned cases, annotation of each instance becomes expensive because it requires hiring people with a high qualification or a specific skill set.

Active learning (AL) is a well-known technique that speeds up data annotation by leveraging model output for selecting instances demonstrated to human experts (Cohn et al., 1996; Settles and Craven, 2008). It focuses human effort on instances that are the most informative for model training, decreasing redundancy and filtering out noisy outliers. AL helps to achieve a certain level of model performance using only a fraction of the labor required to exhaustively annotate a given dataset.

In this work, we present ALToolbox – an open-source framework that contains a comprehensive set of tools for practical AL annotation in text classification, sequence tagging, and seq2seq tasks. The main goal of the framework is to support data scientists and researchers. They usually need to test new ideas very quickly, and the lack of annotation is a common obstacle to this. ALToolbox aims to address several practical obstacles to deploying AL: (1) data annotated with AL should be reusable; (2) AL should not consume excessive computational resources, while the annotation process should be interactive without delays for annotators; (3) annotation should be quick and fluent.

(1) Instances selected with AL that are informative for one model can be not informative for a different model of another type. This hurts the reusability of data annotated with the help of AL. For example, Lowell et al. (2019) shows that if we use predictions of one model for selecting instances during AL, but train a model of a different type on the selected data, the performance of the latter can be even worse compared to the case when it is trained on data labeled without AL. Lowell et al. (2019) call this effect *acquisition-successor mismatch* (ASM) problem (where *acquisition* is a

[1] http://demo.nlpresearch.group
[2] http://demo-video.nlpresearch.group
[3] https://github.com/AIRI-Institute/al_toolbox
♢ Equal contribution

Proceedings of the The 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 406 - 434

Feature	Paladin	ActiveAnno	AlpacaTag	FAMIE	Small-Text	ALToolbox (Ours)
Text classification	✓	✓			✓	✓
Sequence tagging			✓	✓		✓
Seq2seq						✓
SOTA query strategies				✓	✓	✓
SOTA neural models	✓	✓	✓	✓	✓	✓
Computationally efficient AL				✓		✓
Annotated data reusability						✓
Annotation GUI	✓	✓	✓	✓		✓
Serverless annotation in Jupyter						✓
Extensible benchmark						✓
Multilinguality				✓	✓	✓
Compat. with other AL frameworks		✓				✓
Acquisition model adaptation						✓
Proactive learning	✓					

Table 1: Comparison of NLP-related AL frameworks.

model used for selecting instances during AL and *successor* is a model trained on the labeled data for the final application). To address this problem, we include in the framework several pipelines for the preparation of acquisition models and post-processing of data annotated with the help of AL. These pipelines leverage the Pseudo-labeling for the Acquisition-Successor Mismatch (PLASM) algorithm based on the effect of knowledge distillation (Hinton et al., 2015) in AL revealed by Shelmanov et al. (2021); Tsvigun et al. (2022b). PLASM effectively mitigates ASM, making data collected with AL reusable for training models of various architectures.

(2) Applying AL is not free. It introduces additional computational overhead which usually sums up from training an acquisition model and performing its inference. For resource-intensive models such as modern neural networks, this overhead might be prohibitive due to the cost of GPU-accelerated computations for their training and inference. Due to the ASM problem, it is not possible to simply replace a resource-intensive model (e.g. ELECTRA) with a small one (e.g. DistilBERT). PLASM addresses this problem and allows to use small versions of acquisition models obtained using distillation, which speeds up training and inference. ALToolbox also implements an unlabeled pool subsampling algorithm, which leverages uncertainty of instances to avoid repetitive predictions on the part of the unlabeled pool, speeding up the inference phase of AL iterations (Tsvigun et al., 2022b).

(3) AL itself speeds up the annotation procedure, but the time required for deploying an AL-empowered annotation system and integrating annotation with existing data processing pipelines can diminish its benefits. Removing obstacles between the data processing workflow and annotation tools can facilitate rapid evaluation of new ideas. Therefore, in ALToolbox, besides a set of state-of-the-art query strategies, we also provide a serverless AL-empowered annotation tool that is natively integrated directly into the Jupyter Notebook IDE. This tool is suitable for labeling small datasets and testing new ideas quickly, which, we believe, is useful for data scientists and researchers. This tool is easy to start and is fully integrated with the familiar IDE, while also being flexible and extensible.

There are many UI-centric academic and commercial annotation systems for end users that support AL annotation: WebAnno (Yimam et al., 2013), AlpacaTag (Lin et al., 2019), Paladin (Nghiem et al., 2021), ActiveAnno (Wiechmann et al., 2021), FAMIE (Van Nguyen et al., 2022), Prodigy (Montani and Honnibal, 2018) (a commercial system), and others. However, they lack many practical features that serve the goal of rapid annotation, compatibility with pipelines for data analysis and IDEs, and reusability of the annotated data. There are also several low-level AL packages that focus on providing various query strategies and can be used as building blocks for more elaborated systems: LibAct (Yang et al., 2017), ModAL (Danka and Horvath, 2018), Baal (Atighehchian et al., 2020), Small-text (Schröder et al., 2021). However, most of them also overlook the problem of reusability and computational efficiency. Only Small-text is specifically tailored to NLP tasks.

The contributions of the proposed framework:

- a comprehensive collection of state-of-the-art query strategies for sequence tagging, text classification, and seq2seq tasks;
- a benchmarking tool for experimental evaluation of novel AL methods;
- pipelines for acquisition model preparation and for data post-processing that provide reusability of annotated data and computational efficiency of AL;
- a serverless GUI for AL annotation integrated

Method	Paladin	ActiveAnno	AlpacaTag	FAMIE	Small-Text	ALToolbox (ours)
AcTune (Yu et al., 2022)						✓
ALPS (Yuan et al., 2020)				✓		✓
BADGE (Ash et al., 2020)				✓	✓	✓
BAIT (Ash et al., 2021)						✓
BALD (Houlsby et al., 2011)						✓
BatchBALD (Kirsch et al., 2019)						✓
BERT-KM (Yuan et al., 2020)				✓	✓	✓
BLEUVar (Xiao et al., 2020)						✓
Breaking Ties (Luo et al., 2004)					✓	✓
CAL (Margatina et al., 2021)					✓	✓
Cluster-Margin (Citovsky et al., 2021)						✓
Coreset (Sener and Savarese, 2018)					✓	✓
Discriminative AL (Gissin and Shalev-Shwartz, 2019)					✓	
EGL (Settles et al., 2007)					✓	✓
ENSP (Wang et al., 2019)						✓
Entropy (Roy and Mccallum, 2001)					✓	✓
IDDS (Tsvigun et al., 2022a)						✓
LC (Lewis and Gale, 1994)	✓	✓			✓	✓
MNLP (Shen et al., 2017)			✓	✓		✓
NSP (Ueffing and Ney, 2007)						✓
SEALS (Coleman et al., 2022)					✓	

Table 2: The comparison of AL frameworks by implemented query strategies.

directly into the Jupyter notebook IDE for data scientists and researchers.

2 Framework Description

The ALToolbox framework is a Python library with several executable scripts, as well as a Jupyter widget implemented in JavaScript. In this section, we describe the key features of the framework.

2.1 Query Strategies

One of the key components of AL pipelines is a query strategy that specifies what instances are selected for annotation. ALToolbox provides classical and state-of-the-art query strategies for text classification, sequence tagging, and seq2seq tasks. Table 2 summarizes strategies implemented in our framework and in software packages from the related work.

Uncertainty sampling is one of the most widely-used types of AL query strategies. ALToolbox provides several basic uncertainty estimation methods based on softmax prediction probability: **Least Confidence (LC)** (Lewis and Gale, 1994), **Maximum Normalized Log-Probability (MNLP)** (Shen et al., 2017), **Breaking Ties (BT)** (Luo et al., 2004), **Prediction entropy (PE)** (Roy and Mccallum, 2001), **Normalized Sequence Probability (NSP)** (Ueffing and Ney, 2007). Since a predictive distribution of a single deterministic neural network cannot be used to obtain reliable uncertainty scores (Sener and Savarese, 2018; Mukhoti et al., 2021), some works have ventured into the development of Bayesian query strategies (Siddhant and Lipton, 2018). ALToolbox implements one of the widely-adopted strategies – **Bayesian Active Learning by Disagreement (BALD)** (Houlsby et al., 2011). It selects instances that provide the biggest amount of information about true model parameters by knowing the true label of the considered instance. In practice, the strategy approximates variational inference in a Bayesian neural network using Monte-Carlo dropout (Gal and Ghahramani, 2016). ALToolbox also includes a batched version of BALD – **BatchBALD** (Kirsch et al., 2019), which is modified to jointly score and select for annotation multiple instances on each AL iteration.

An alternative for uncertainty sampling is diversity-based sampling. In this category, the **coreset** algorithm (Sener and Savarese, 2018) leverages data geometry and aims to minimize the bound between an average loss over any given subset of the dataset and the remaining data points. Recently proposed **Contrastive Active Learning (CAL)** prioritizes instances, which predictive likelihoods diverge the most from their neighbors in the training set (Margatina et al., 2021). The **Cluster-Margin** algorithm (Citovsky et al., 2021) is designed to select large batches for annotation. It prioritizes instances that are diverse and that the model is not confident about. **BERT-KM** (Yuan et al., 2020) clusters texts in the unlabeled pool using their contextualized embeddings and selects the nearest neighbors of cluster centers. **Active Learning by Processing Surprisal (ALPS)** (Yuan et al., 2020) leverages pre-trained models, self-supervised learning objective, and clustering to solve the cold-start problem in AL. **AcTune** (Yu et al., 2022) can be used as a wrapper over uncertainty-based query strategies. It selects the most uncertain instances from regions obtained by clustering the unlabeled pool and ranking them by uncertainty and diversity.

ALToolbox also contains several gradient-based

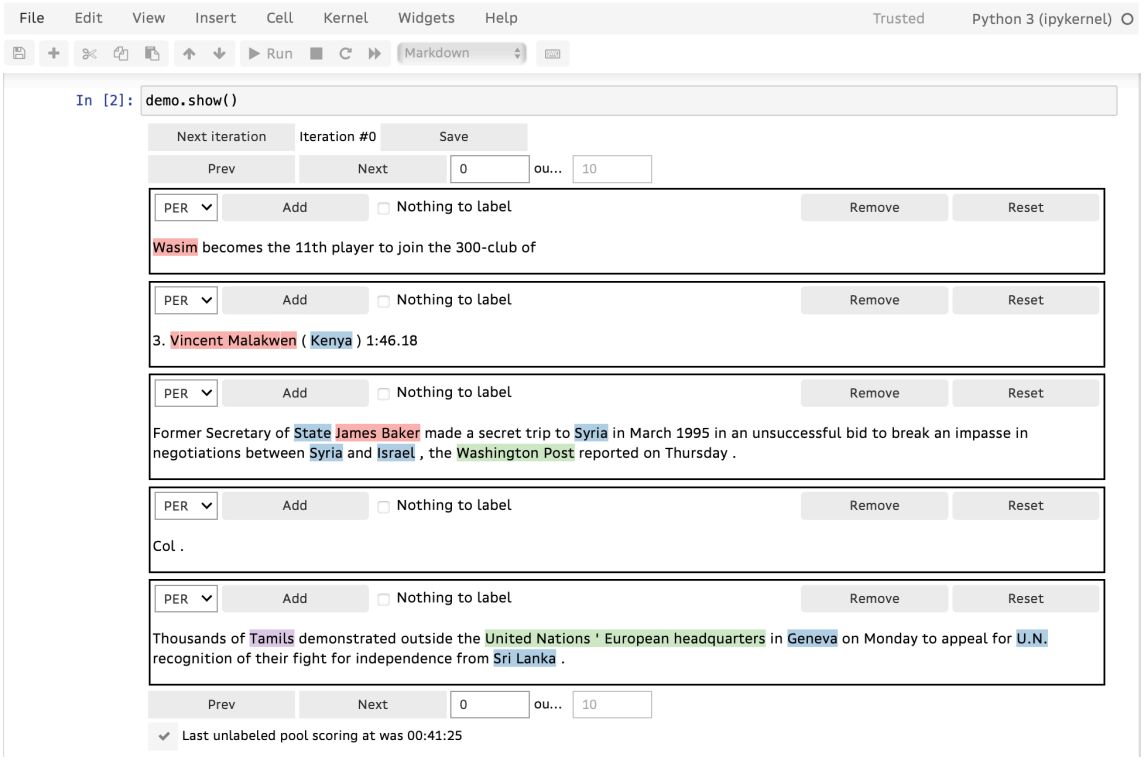


Figure 1: Serverless GUI annotation tool integrated into the Jupyter IDE.

query strategies. **Expected Gradient Length (EGL)** aims to prioritize instances that would impart the greatest change to the current model if we add them to the training set with their labels (Settles et al., 2007). **Batch Active Learning by Diverse Gradient Embeddings (BADGE)** measures uncertainty as the gradient magnitude with respect to parameters in the final (output) layer (Ash et al., 2020). **Batch Active learning via Information maTrices (BAIT)** selects batches of instances by optimizing a bound on the MLE error in terms of the Fisher information (Ash et al., 2021).

Furthermore, ALToolbox provides several query strategies for seq2seq tasks. **NSP** (Ueffing and Ney, 2007) is an analogue of LC for text generation, which calculates the length-normalized total probability of a generated sequence. **ENSP** (Wang et al., 2019) makes several stochastic runs using Monte-Carlo dropout and averages the probabilities of the sequences. The **BLEUVar** (Xiao et al., 2020) algorithm strives to measure the variance of texts generated under Monte-carlo dropout by using the BLEU metric (Papineni et al., 2002). The **IDDS** (Tsvigun et al., 2022a) strategy, shown to be state-of-the-art for the abstractive text summarization task, selects instances that are semantically dissimilar from the already annotated instances, avoiding outliers and borderline instances.

Finally, the framework provides the ability to use different strategies for different AL iterations. For example, one could use a cold-start method (e.g. ALPS) at several first iterations and later switch to another strategy such as LC.

2.2 Supported Models

ALToolbox is compatible with the HuggingFace Transformers library (Wolf et al., 2020), allowing the usage of state-of-the-art Transformer models like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), XLNet (Yang et al., 2019), ELECTRA (Clark et al., 2020), and others. The support of some older RNN-based models like CNN-BiLSTM-CRF (Ma and Hovy, 2016) for sequence tagging is implemented via a wrapper around the Flair library (Akbik et al., 2019). Users can also implement their own models directly using PyTorch. ALToolbox provides several custom neural model implementations in PyTorch, including the classical CNN for text classification (Le et al., 2018).

2.3 Jupyter Annotation Tool

ALToolbox provides a simple serverless tool with a GUI for AL annotation integrated directly into Jupyter Notebook, which is one of the most popular IDEs for the Python language and data analysis (Figure 1). It supports annotation for text classification and sequence tagging tasks like named entity

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Average
Actune + CAL	92.0 ± 0.5	94.3 ± 0.0	**94.8 ± 0.2**	**95.0 ± 0.2**	**94.4 ± 0.0**
Actune + Entropy	91.7 ± 0.5	94.1 ± 0.3	94.7 ± 0.4	94.8 ± 0.3	**94.2 ± 0.2**
Actune + LC	92.0 ± 0.3	94.2 ± 0.2	**94.7 ± 0.4**	95.0 ± 0.0	**94.3 ± 0.3**
ALPS	90.6 ± 0.3	92.3 ± 0.5	92.9 ± 0.5	93.4 ± 0.6	92.6 ± 0.2
BADGE	**92.2 ± 0.3**	94.3 ± 0.1	94.7 ± 0.2	95.0 ± 0.2	**94.3 ± 0.1**
BAIT	**92.2 ± 0.4**	94.3 ± 0.1	**95.0 ± 0.2**	**95.1 ± 0.4**	**94.4 ± 0.1**
BALD	**92.2 ± 0.6**	93.9 ± 0.1	**94.8 ± 0.4**	**95.2 ± 0.1**	**94.3 ± 0.2**
Breaking Ties (BT)	**92.3 ± 0.3**	**94.5 ± 0.3**	94.9 ± 0.1	95.0 ± 0.0	**94.4 ± 0.1**
CAL	**92.3 ± 0.7**	**94.5 ± 0.3**	**94.8 ± 0.3**	**95.1 ± 0.1**	**94.4 ± 0.2**
Coreset	91.9 ± 0.1	93.8 ± 0.3	**94.7 ± 0.5**	**95.1 ± 0.1**	94.2 ± 0.1
Entropy	92.0 ± 0.1	94.4 ± 0.1	94.9 ± 0.1	**95.0 ± 0.2**	**94.4 ± 0.1**
Least Confidence (LC)	**92.3 ± 0.5**	**94.3 ± 0.3**	**95.0 ± 0.4**	**95.0 ± 0.2**	**94.4 ± 0.1**
Mahalanobis Distance	91.6 ± 0.4	94.0 ± 0.3	94.8 ± 0.2	95.0 ± 0.1	94.2 ± 0.1
Random	90.8 ± 0.4	92.5 ± 0.1	92.9 ± 0.4	93.4 ± 0.5	92.6 ± 0.2

Table 3: Accuracy of RoBERTa on AG News with various AL strategies on several AL iterations with query size = 1% (1200 instances). *Average* refers to the average result throughout the AL cycle. We select with **bold** state-of-the-art results with respect to confidence intervals. The results are averaged for 5 runs with different seeds to ensure the stability.

recognition and event extraction.

The tool is implemented using Jupyter widgets – a built-in feature of the Jupyter IDE for creating extensions. This widget can be configured with various AL query strategies and models, including Transformers. After the tool object is invoked, the IDE displays the widget in a notebook cell, and AL annotation begins. For example, to add NER annotation, a user can select a corresponding text fragment with a mouse and add a label to it. For text classification, a label can be chosen from a predefined list via selectable buttons. On each iteration, the user receives instances for annotation in mini-batches. The user can annotate all or just a part of them and invoke the next iteration of an AL algorithm with the "Next iteration" button asking for a new minibatch of unlabeled instances.

The annotation tool performs all necessary computations asynchronously with GUI and returns new instances without any delay. It keeps a list of instances sorted by their "informativeness" and updates it as soon as possible in the background.

The user can interrupt annotation at any time and resume it after a while. The tool tracks changes made by the user on the hard drive. The annotation is accumulated in easy-to-parse JSON files.

The target audience of this tool is data scientists and researchers. It is very easy to launch and modify: new graphical elements can be added using Jupyter Widgets as well. Using Jupyter also helps to reduce the effort of combining the system with data processing pipelines. We consider this tool might be useful for rapid annotation in small to medium projects and for testing new ideas quickly. However, we note that it lacks many useful features of full-fledged annotation systems, e.g., the ability to work with multiple users simultaneously. Creating a complex GUI for annotation is out of the scope of this project since a wide range of similar projects have already been released, e.g. DocAnno (Nakayama et al., 2018), LabelStudio (Tkachenko et al., 2020-2022), ActiveAnno (Wiechmann et al., 2021). The ALToolbox framework can be easily integrated into such annotation systems with the help of API.

2.4 Tools for Computational Efficient Active Learning and Reusable Annotation

ALToolbox contains a set of scripts that help to improve the computational efficiency of AL while keeping annotated data reusable. AL requires a substantial amount of computations on each iteration, which depends on the complexity and the size of the acquisition model. Using smaller and lighter models can lead to performance degradation of AL due to the ASM problem discussed in the introduction. We mitigate this problem by implementing tools for the "Pseudo-Labeling for Acquisition-Successor Mismatch" (PLASM) algorithm (Tsvigun et al., 2022b). This algorithm leverages small distilled models (e.g. DistilBERT) during the acquisition of instances, but after the annotation is finished it trains the original full-sized models (e.g. BERT) on the acquired data and uses it for automatic pseudo-labeling of the whole unlabeled pool of instances. The mistakes in automatic annotation are cleaned with the help of the TracIn method (Pruthi et al., 2020). Finally, the successor model is trained on the data that contains gold-standard labels and cleaned automatically labeled instances.

PLASM reduces or completely removes the gap in performance that appears when the successor model is different from the acquisition model. It makes the annotated data reusable for training suc-

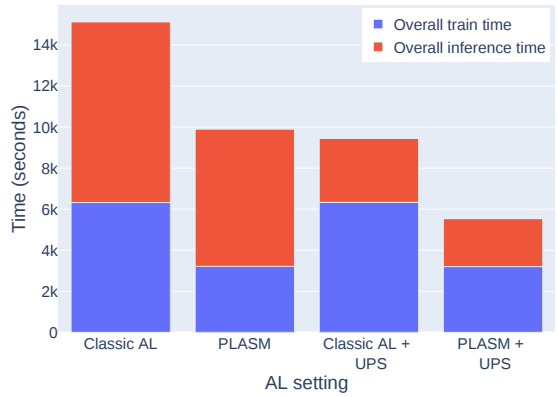


Figure 2: Duration in seconds of all the training and inference phases of the simulated AL with different acquisition settings on AG News with query size = 1% and 15 AL iterations. ELECTRA is used as a successor model, and DistilBERT – for acquisition in PLASM.

cessor models of various architectures. ALToolbox provides scripts for automatic model distillation and a pipeline for data post-processing with PLASM. All the necessary post-processing can be done by invoking a single function.

For large datasets, making predictions for the whole unlabeled set on each iteration to obtain the uncertainty estimates may require an enormous amount of time and resources. Consequently, in the framework, we also implement the "unlabeled pool subsampling" (UPS) algorithm (Tsvigun et al., 2022b), which samples the instances from the unlabeled pool according to their uncertainty estimates on previous iterations.

Figures 2, 21, 23 illustrate time benefits brought by PLASM and UPS on AG News, IMDB, and CoNLL-2003, respectively. PLASM accelerates the training phase of AL by 35% / 65% / 34%, while UPS accelerates the inference phase by 65% / 61% / 63%. Their combination speeds up all AL iteration computations by up to 63% / 67% / 38%, respectively, making AL much more interactive. Figures 19a, 20a, 22a show that the performance of the successor model does not deteriorate when these algorithms are used. Figures 19b, 20b, 22b, in turn, show that the ASM problem leads to a substantial decrease in the model performance.

We also provide scripts for domain adaptation of acquisition models. Margatina et al. (2022) demonstrate that self-supervised adaptation (Gururangan et al., 2020) of pre-trained Transformers on the unlabeled pool of instances helps to speed up AL.

2.5 Benchmarking Tool for Query Strategies

ALToolbox provides an extensible and easy-to-use benchmarking tool for testing new AL query strategies and unlabeled pool subsampling strategies. To experiment with a new strategy, a user implements it in the form of a Python class and runs the evaluation script, specifying the path to the corresponding class module as an argument. The script performs several iterations of simulated AL annotation and constructs the dependence of the model performance scores on the size of the labeled data. Experiments are launched multiple times with different random seeds to obtain confidence intervals of the results.

Using this tool, we provide the evaluation results of implemented query strategies, which can be used as a reference. The experiments with text classification are conducted on AG News (Zhang et al., 2015), IMDB (Maas et al., 2011), and CoLA (Warstadt et al., 2018); with sequence tagging – on CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003); with abstractive text summarization – on AESLC (Zhang and Tetreault, 2019), WikiHow (Koupaee and Wang, 2018), and PubMed (Cohan et al., 2018). We provide the results with big and lightweight Transformers and with several different query sizes:

- Selecting $k\%$ of instances (for text classification & abstractive text summarization) / tokens (for sequence tagging). In this setting, we randomly select and annotate $k\%$ of instances / tokens as the initial seed and select $k\%$ of instances / tokens for annotation on each AL iteration according to the query function. This configuration aims to benchmark strategies in a high-resource AL mode. We refer to it as *query size* = $k\%$.
- Selecting 100 instances / tokens on each AL iteration and as the initial seed. This configuration aims to benchmark strategies in a medium-resource AL mode. We refer to it as *query size = 100*.
- Selecting 10 instances / tokens on each AL iteration. The initial seeding procedure differs between tasks under this mode. For text classification, we randomly select and annotate 1 instance of each class as the initial seed. For other tasks, we annotate 10 randomly chosen instances / tokens. This configuration aims to benchmark strategies in a low-resource AL mode. We refer to it as *query size = 10*.

Dataset statistics, model details, and hyperparameters are presented in Tables 4–6.

Table 3 depicts the results on AG News with RoBERTa-base as an acquisition model, and query size = 1%. We can see that most of the strategies perform roughly similar with CAL and LC showing the best performance across all AL iterations. Figure 3 also demonstrates the results throughout the whole AL cycle of the best-performing query strategies according to the average accuracy throughout the AL cycle. Figure 4 provides the comparison of the duration of computations for various query strategies. Tables 7–18 compare query strategies on text classification datasets for various settings and models. Figures 5–10 visualize the results of the best-performing query strategies.

Sequence tagging results are presented in Tables 19–23. MNLP demonstrates the best quality in terms of F1-micro score excluding the "no entity" tag ("O"). Figures 11–13 show the iteration-wise scores. The duration of computations for various strategies is presented in Figure 23.

For abstractive text summarization, due to the big size of the unlabeled pool of WikiHow and Pubmed, on each AL iteration, we randomly subsample the unlabeled pool to 10,000 instances. Tables 24–27 provide the average results throughout the AL cycle and results on several iterations, while Figures 15–18 illustrate the results during the entire AL cycle. Finally, Figure 14 compares the duration of execution of the seq2seq query strategies.

3 Related Work

The comparison of ALToolbox with other frameworks from the related work on AL in NLP is presented in Table 1.

First of all, ALToolbox supports two most demanded NLP tasks: text classification and sequence tagging. It also works with abstractive text summarization, which is a seq2seq task. Other frameworks support only one of the tasks: Paladin, ActiveAnno, and Small-Text work only with text classification, while AlpacaTag and FAMIE support only sequence tagging.

Table 2 compares AL frameworks by implemented query strategies. Paladin, ActiveAnno, and AlpacaTag implement only the basic strategies. FAMIE implements several modern methods like ALPS and BADGE, but lacks many others. We note that Small-Text implements many recently proposed query strategies, including CAL, BADGE, and BERT-KM. However, ALToolbox provides the most comprehensive set of state-of-the-art query strategies and also allows combining them.

Except for ALToolbox and FAMIE (Van Nguyen et al., 2022), the computational overhead and the AL-caused time delays have been inexplicably dismissed in the prior art. FAMIE entails training a bigger model in the background during the labeling of each batch while using a smaller one as a proxy for acquisition. Such knowledge distillation makes the AL annotation process more interactive but also carries an additional computational burden, requiring extra resources for training and running two models. On the contrary, the knowledge distillation within our framework reduces both the time needed to complete an AL iteration and the overall amount of computation.

We note that neither FAMIE, nor other frameworks, address the ASM problem that hinders the reusability of annotated data. The tools for model distillation and annotated data post-processing based on the PLASM algorithm in our framework help to mitigate the ASM, so a user, for example, can train XLNet using data acquired with DistilBERT without significant performance penalties.

Most of the considered systems provide an elaborated GUI for annotation by end-users. Our framework aims to support data scientists and researchers and provides a fast-to-deploy minimalistic annotation system directly in the Jupyter IDE.

None of the considered systems provides easy-to-use scripts for conducting experiments with new AL methods. ALToolbox implements an extensible benchmarking tool that we hope will simplify research in AL for NLP.

One of the problems that are currently out of the scope of ALToolbox is efficient task assignments to multiple annotators. Proactive learning implemented in Paladin addresses this problem. We consider this feature as future work.

4 Conclusion

We introduced ALToolbox, an open-source framework for practical AL in NLP. Besides many other features, the framework addresses the problems of computational efficiency of AL and data reusability. We hope that our framework will foster the development of new AL methods and remove some practical obstacles to deploying AL annotation.

In future work, we are looking forward to adding the support of more text generation tasks, introducing proactive learning, and providing tools for hyperparameter selection in AL.

Acknowledgements

We thank anonymous reviewers for their insightful suggestions to improve this paper. The work was supported by the Russian Science Foundation grant 20-71-10135 (all sections except Section 2.4). The work by Olga Shakhmatova and Dmitry V. Dylov on Section 2.4 and the GUI annotation tool is supported by the grant of RFBR #19-29-01240. The experiments were supported in part by computational resources of HPC facilities at HSE University (Kostenetskiy et al., 2021).

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Jordan Ash, Surbhi Goel, Akshay Krishnamurthy, and Sham Kakade. 2021. Gone fishing: Neural active learning with fisher embeddings. In *Advances in Neural Information Processing Systems*, volume 34, pages 8927–8939. Curran Associates, Inc.

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*.

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*.

Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. 2021. Batch active learning at scale.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. 1996. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145.

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*, 36(6):6402–6410.

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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA. PMLR.

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

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

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

Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. 2019. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32.

P. S. Kostenetskiy, R. A. Chulkevich, and V. I. Kozyrev. 2021. HPC Resources of the Higher School of Economics. *Journal of Physics: Conference Series*, 1740(1):012050.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *CoRR*, abs/1810.09305.

Hoa T Le, Christophe Cerisara, and Alexandre Denis. 2018. Do convolutional networks need to be deep for text classification? In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer.

Bill Yuchen Lin, Dong-Ho Lee, Frank F. Xu, Ouyu Lan, and Xiang Ren. 2019. AlpacaTag: An active learning-based crowd annotation framework for sequence tagging. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 58–63, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

David Lowell, Zachary C. Lipton, and Byron C. Wallace. 2019. Practical obstacles to deploying active learning. 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 21–30, Hong Kong, China. Association for Computational Linguistics.

Tong Luo, K. Kramer, S. Samson, A. Remsen, D.B. Goldgof, L.O. Hall, and T. Hopkins. 2004. Active learning to recognize multiple types of plankton. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 478–481 Vol.3.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150. The Association for Computer 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.

Ines Montani and Matthew Honnibal. 2018. Prodigy: A new tool for radically efficient machine teaching. `https://prodi.gy/`.

Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip HS Torr, and Yarin Gal. 2021. Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. *arXiv preprint arXiv:2102.11582*.

Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. 2018. doccano: Text annotation tool for human. Software available from https://github.com/doccano/doccano.

Minh-Quoc Nghiem, Paul Baylis, and Sophia Ananiadou. 2021. Paladin: an annotation tool based on active and proactive learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 238–243, Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Nicholas Roy and Andrew Mccallum. 2001. Toward optimal active learning through sampling estimation of error reduction. *Proceedings of the 18th International Conference on Machine Learning*.

Christopher Schröder, Lydia Müller, Andreas Niekler, and Martin Potthast. 2021. Small-text: Active learning for text classification in python.

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

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25-27 October 2008, Honolulu,*

Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 1070–1079. Association for Natural Language Processing.

Burr Settles, Mark Craven, and Soumya Ray. 2007. Multiple-instance active learning. *Advances in neural information processing systems*, 20.

Artem Shelmanov, Dmitri Puzyrev, Lyubov Kupriyanova, Denis Belyakov, Daniil Larionov, Nikita Khromov, Olga Kozlova, Ekaterina Artemova, Dmitry V. Dylov, and Alexander Panchenko. 2021. Active learning for sequence tagging with deep pre-trained models and Bayesian uncertainty estimates. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1698–1712, Online. Association for Computational Linguistics.

Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*.

Aditya Siddhant and Zachary C. Lipton. 2018. Deep Bayesian active learning for natural language processing: Results of a large-scale empirical study. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2904–2909, Brussels, Belgium. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. 2020-2022. Label Studio: Data labeling software. Open source software available from https://github.com/heartexlabs/label-studio.

Akim Tsvigun, Ivan Lysenko, Danila Sedashov, Ivan Lazichny, Eldar Damirov, Vladimir Karlov, Artemy Belousov, Leonid Sanochkin, Maxim Panov, Alexander Panchenko, Mikhail Burtsev, and Artem Shelmanov. 2022a. Active learning for abstractive text summarization. In *Findings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

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

Nicola Ueffing and Hermann Ney. 2007. Word-level confidence estimation for machine translation. *Comput. Linguistics*, 33(1):9–40.

Minh Van Nguyen, Nghia Trung Ngo, Bonan Min, and Thien Huu Nguyen. 2022. Famie: A fast active learning framework for multilingual information extraction.

Shuo Wang, Yang Liu, Chao Wang, Huanbo Luan, and Maosong Sun. 2019. Improving back-translation with uncertainty-based confidence estimation. 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 791–802, Hong Kong, China. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Max Wiechmann, Seid Muhie Yimam, and Chris Biemann. 2021. ActiveAnno: General-purpose document-level annotation tool with active learning integration. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 99–105, Online. Association for Computational Linguistics.

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*, pages 38–45, Online. Association for Computational Linguistics.

Tim Z. Xiao, Aidan N. Gomez, and Yarin Gal. 2020. Wat zei je? detecting out-of-distribution translations with variational transformers. *CoRR*, abs/2006.08344.

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*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. Webanno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st*

Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 1–6.

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.

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.

Rui Zhang and Joel Tetreault. 2019. This email could save your life: Introducing the task of email subject line generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 446–456, Florence, Italy. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

A Dataset Statistics and Model Hyperparameters

Dataset	Train	Test	Mean document length (tokens)	C
CoNLL-2003	15K/203.6K	3.7K/46.4K	14.5	4(5)
AG News	120K	7.6K	53.1	4
IMDb	25K	25K	300.0	2
CoLA	8.5K	1K	11.3	2
AESLC	14.4K	1.9K	142.4	-
WikiHow	184.6K	1K	377.5	-
Pubmed	119.1K	6.7K	495.4	-

Table 4: Dataset statistics. We provide a number of instances / tokens (for sequence tagging) for the training and test sets and average lengths of documents in terms of tokens. C is a number of classes / entity types for text classification and sequence tagging datasets.

Task	Model	Checkpoint	# Param.
Text classification	BERT	bert-base-uncased	110M
	DistilBERT	distilbert-base-uncased	67M
	ELECTRA	google/electra-base-discriminator	110M
	DistilELECTRA	lsanochkin/distilelectra-base	67M
	RoBERTa	roberta-base	125M
	DistilRoBERTa	distilroberta-base	82M
Sequence tagging	ELECTRA	google/electra-base-discriminator	110M
	BERT	bert-base-cased	110M
	DistilBERT	distilbert-base-cased	67M
Abstractive text summarization	BART	facebook/bart-base	139M
	PEGASUS	google/pegasus-large	570M

Table 5: Transformers model checkpoints from the HuggingFace repository (Wolf et al., 2020)

Hparam	Sequence tagging	Classification	BART	PEGASUS
Number of epochs	15	5	6	4
Batch size	16	16	16	2
Gradient accumulation steps	1	1	1	8
Min. number of training steps	1000	1000	350	200
Max. sequence length	-	256	1024	1024
Optimizer	AdamW			
Learning rate	5e-5	2e-5	2e-5	5e-4
Weight decay	0.01	0.01	0.028	0.03
Gradient clipping	1.	1.	0.28	0.3
Sheduler	STLR			
% warm-up steps	10			
Num. beams at evaluation	-	-	4	4
Generation max. length	-	-	Adapt.	Adapt.

Table 6: Hyperparameter values of Transformers. The hyperparameters are chosen according to evaluation scores on the validation datasets when models are trained using the whole available training data. *Adapt* refers to adaptive length, when generation maximum length is equal to the maximum summary length on the train set.

B Query Strategy Benchmark

For the tables in this section, we select with **bold** state-of-the-art results with respect to the confidence intervals. When all the values are within the confidence interval, we only select with **bold** the largest average value. The results are averaged for 10 runs with different seeds for *query size = 10* and for 5 runs for other query size settings to ensure stability. The *Average* column refers to the average result throughout the AL cycle.

B.1 Text Classification

B.1.1 AG News

Query size = 1 %

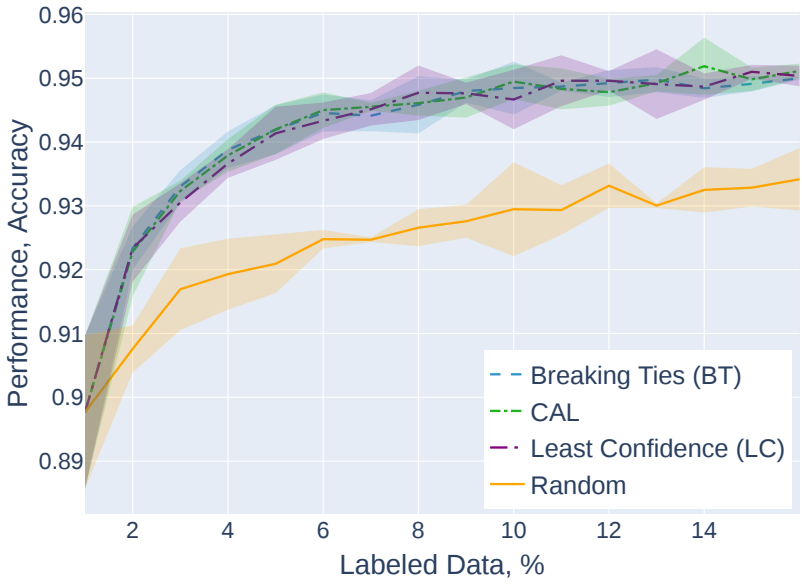


Figure 3: Accuracy of the best performing query strategies according to average accuracy throughout the AL cycle (BT, CAL, and LC) on AG News with RoBERTa with query size = 1%.

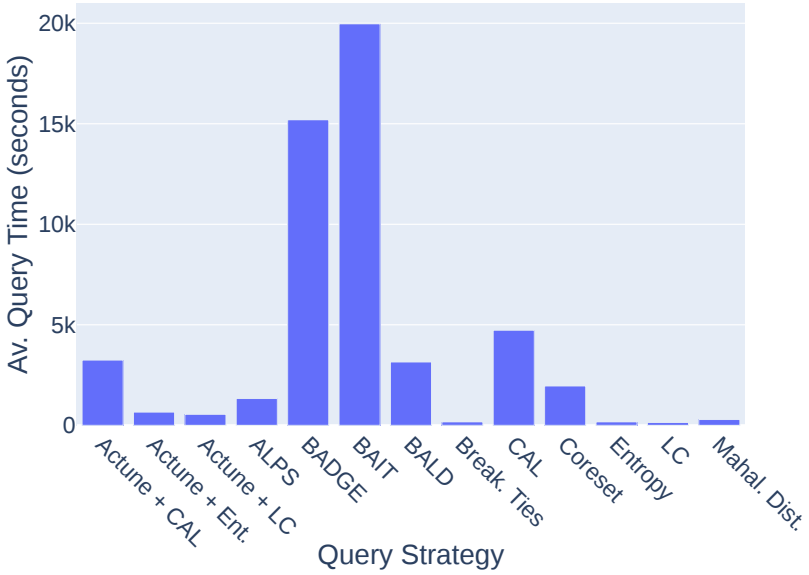
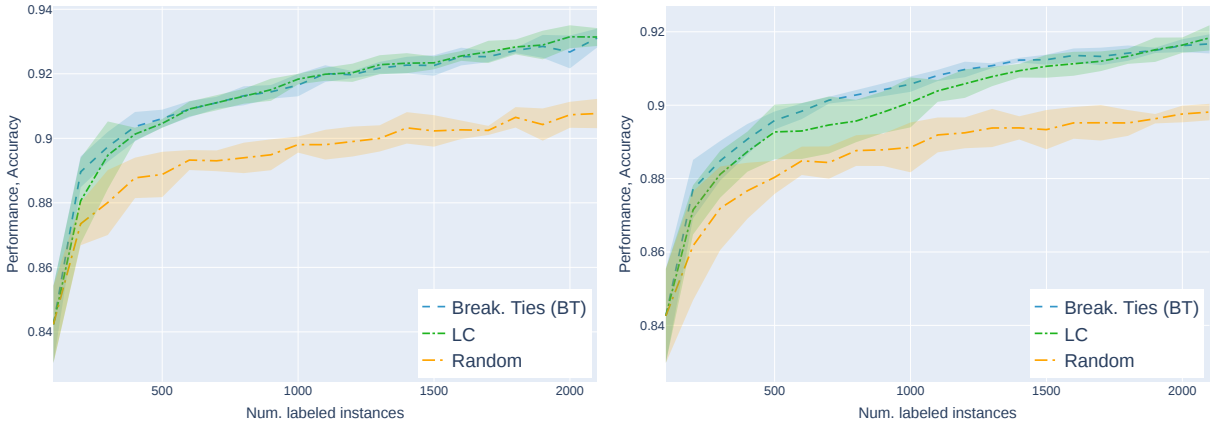


Figure 4: Average duration in seconds of one AL query with different strategies on AG News with RoBERTa as an acquisition model and query size = 1% (1200 instances). Hardware configuration is provided in Appendix C.

Query size = 100



a) RoBERTa b) DistilBERT

Figure 5: Accuracy of the best performing query strategies with different acquisition models on AG News with query size = 100.

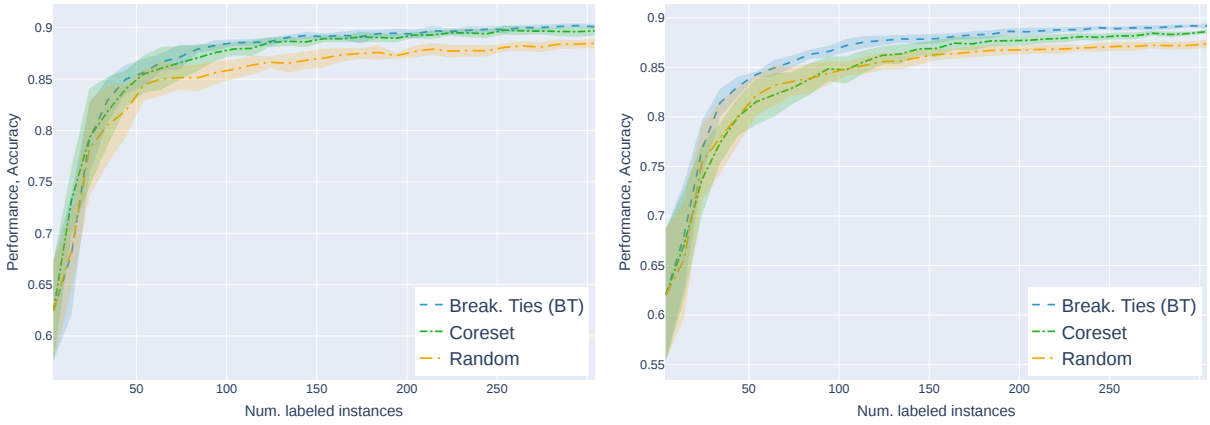
AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Average
BADGE	**88.95 ± 0.85**	**90.95 ± 0.18**	91.83 ± 0.2	**92.49 ± 0.27**	**93.06 ± 0.25**	**91.7 ± 0.16**
BAIT	88.21 ± 1.36	90.56 ± 0.28	91.57 ± 0.28	92.23 ± 0.31	92.87 ± 0.15	91.32 ± 0.36
BALD	88.24 ± 1.04	90.0 ± 0.74	90.98 ± 0.29	91.26 ± 0.46	91.98 ± 0.32	90.69 ± 0.39
BT	**88.96 ± 0.48**	**90.9 ± 0.25**	**92.01 ± 0.26**	**92.53 ± 0.28**	**93.11 ± 0.29**	**91.66 ± 0.06**
CAL	88.08 ± 0.93	90.67 ± 0.15	91.73 ± 0.31	**92.42 ± 0.17**	92.91 ± 0.23	91.48 ± 0.16
Coreset	87.97 ± 1.26	90.32 ± 0.42	91.33 ± 0.24	91.72 ± 0.17	92.23 ± 0.32	90.97 ± 0.26
Entropy	88.02 ± 1.14	90.65 ± 0.35	91.24 ± 0.38	91.95 ± 0.42	92.58 ± 0.3	91.15 ± 0.27
LC	88.06 ± 1.33	**90.91 ± 0.23**	**91.99 ± 0.2**	**92.55 ± 0.15**	**93.14 ± 0.28**	**91.65 ± 0.15**
Random	87.35 ± 0.66	89.33 ± 0.31	89.8 ± 0.46	90.26 ± 0.28	90.77 ± 0.45	89.68 ± 0.27

Table 7: Accuracy of RoBERTa on AG News with various AL strategies with query size = 100.

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Average
BALD	86.57 ± 0.26	89.06 ± 0.49	90.17 ± 0.21	90.57 ± 0.17	90.99 ± 0.29	89.82 ± 0.23
BT	**87.71 ± 0.8**	**89.84 ± 0.22**	**90.81 ± 0.15**	**91.36 ± 0.19**	**91.67 ± 0.26**	**90.52 ± 0.12**
CAL	87.37 ± 0.34	**89.43 ± 0.37**	90.37 ± 0.34	91.11 ± 0.22	**91.59 ± 0.26**	90.2 ± 0.2
Coreset	86.84 ± 0.59	89.31 ± 0.34	89.96 ± 0.23	90.51 ± 0.36	91.1 ± 0.21	89.81 ± 0.28
Entropy	86.27 ± 0.57	89.15 ± 0.63	90.05 ± 0.28	90.65 ± 0.55	91.19 ± 0.23	89.81 ± 0.37
LC	**87.15 ± 0.67**	**89.3 ± 0.76**	**90.39 ± 0.3**	91.13 ± 0.33	**91.83 ± 0.34**	90.19 ± 0.31
Random	86.17 ± 1.48	88.48 ± 0.39	89.19 ± 0.48	89.52 ± 0.43	89.81 ± 0.23	88.83 ± 0.4

Table 8: Accuracy of DistilBERT on AG News with various AL strategies with query size = 100.

Query size = 10



a) RoBERTa-base b) DistilBERT-base

Figure 6: Accuracy of the best performing query strategies with different acquisition models on AG News with query size = 10.

AL Strat.	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Iter. 25	Iter. 30	Average
BADGE	**68.47 ± 7.84**	**85.13 ± 1.43**	**88.2 ± 0.58**	**88.68 ± 0.62**	**89.38 ± 0.25**	**89.86 ± 0.24**	89.8 ± 0.18	**87.3 ± 1.59**
BALD	60.0 ± 6.97	79.58 ± 3.13	84.34 ± 1.4	85.64 ± 1.63	86.48 ± 1.1	87.37 ± 1.18	87.61 ± 1.13	83.77 ± 1.23
BT	67.98 ± 5.91	**85.64 ± 1.31**	**88.56 ± 0.32**	**89.13 ± 0.32**	**89.39 ± 0.41**	**89.83 ± 0.36**	**90.1 ± 0.26**	**87.66 ± 0.51**
CAL	58.04 ± 6.22	68.89 ± 7.78	86.33 ± 2.47	87.43 ± 1.47	88.58 ± 0.84	88.81 ± 0.57	89.38 ± 0.37	83.23 ± 1.77
Coreset	**73.21 ± 3.27**	**85.49 ± 1.82**	87.93 ± 0.63	**88.94 ± 0.36**	**89.29 ± 0.31**	**89.77 ± 0.46**	**89.69 ± 0.43**	**87.44 ± 0.8**
Entropy	60.64 ± 8.75	82.29 ± 1.72	86.45 ± 0.5	87.33 ± 0.8	88.45 ± 0.6	89.25 ± 0.49	89.32 ± 0.61	85.57 ± 0.83
LC	61.86 ± 8.49	**85.36 ± 0.88**	87.38 ± 0.51	88.47 ± 0.59	88.99 ± 0.38	89.35 ± 0.27	89.65 ± 0.23	86.6 ± 0.25
Random	68.33 ± 3.88	84.41 ± 1.52	85.95 ± 0.94	87.05 ± 0.92	87.68 ± 0.57	88.09 ± 0.39	88.47 ± 0.47	85.73 ± 0.69

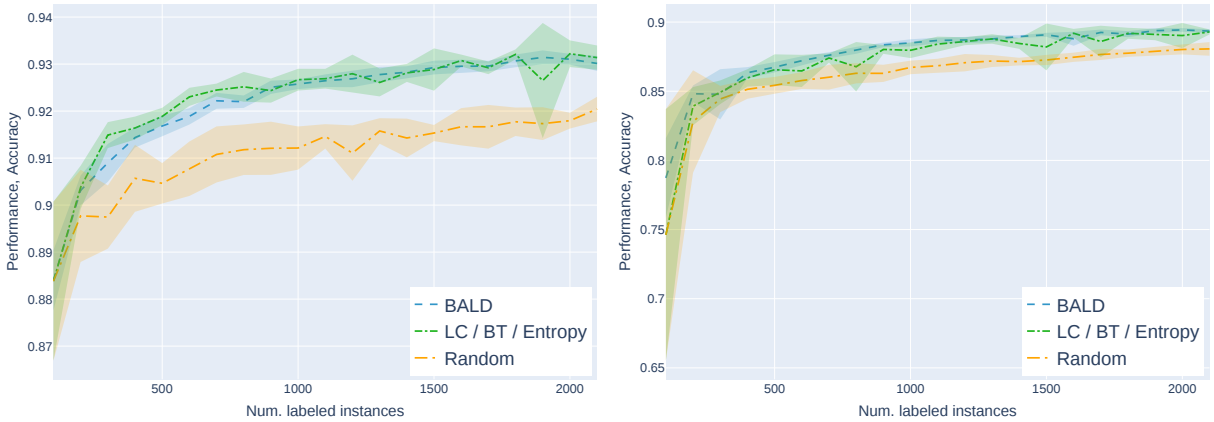
Table 9: Accuracy of RoBERTa on AG News with various AL strategies with query size = 10.

AL Strat.	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Iter. 25	Iter. 30	Average
BT	**67.95 ± 5.39**	**84.27 ± 0.3**	**87.2 ± 0.66**	**87.9 ± 0.51**	**88.59 ± 0.39**	**88.91 ± 0.19**	**89.19 ± 0.26**	**86.48 ± 0.31**
CAL	58.87 ± 6.36	70.09 ± 4.32	82.72 ± 2.05	85.72 ± 1.17	87.29 ± 0.67	87.68 ± 0.78	88.31 ± 0.48	81.84 ± 1.64
Coreset	**67.07 ± 5.52**	81.52 ± 2.31	84.74 ± 1.37	86.91 ± 0.87	87.72 ± 0.49	88.17 ± 0.34	88.61 ± 0.24	85.0 ± 0.84
Entropy	56.19 ± 9.83	80.54 ± 2.05	84.65 ± 1.15	85.97 ± 0.89	86.48 ± 1.12	87.21 ± 0.55	87.5 ± 0.61	83.47 ± 0.89
LC	54.96 ± 4.34	82.28 ± 1.18	85.33 ± 0.9	86.99 ± 0.75	87.86 ± 0.39	88.38 ± 0.24	88.57 ± 0.38	84.76 ± 0.49
Random	65.56 ± 5.91	82.14 ± 2.01	84.87 ± 0.69	86.29 ± 0.58	86.77 ± 0.43	87.11 ± 0.44	87.37 ± 0.42	84.46 ± 0.91

Table 10: Accuracy of DistilBERT on AG News with various AL strategies with query size = 10.

B.1.2 IMDB

Query size = 100



a) RoBERTa b) DistilBERT

Figure 7: Accuracy of the best performing query strategies with different acquisition models on IMDB with query size = 100.

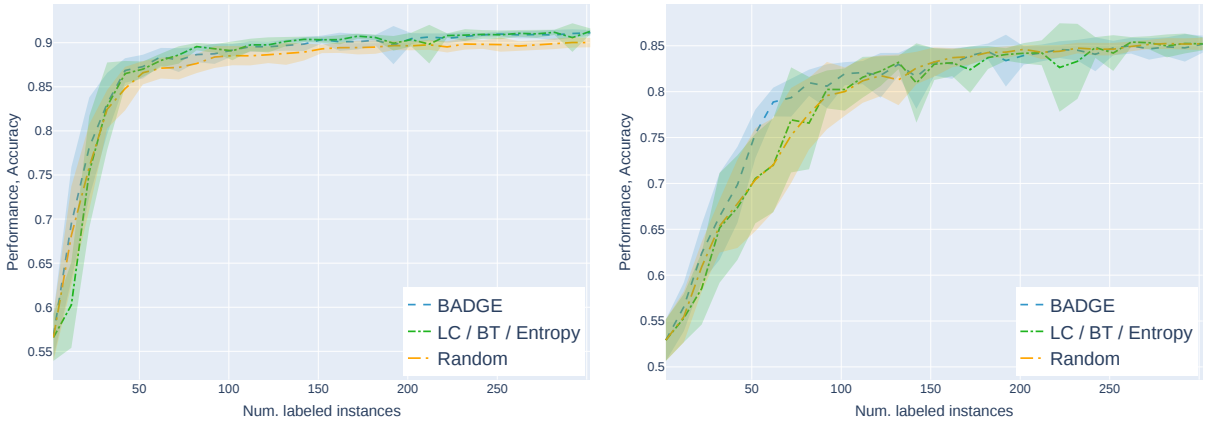
AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Average
BADGE	89.66 ± 1.17	91.91 ± 0.31	**92.32 ± 0.59**	92.69 ± 0.4	**93.22 ± 0.17**	92.35 ± 0.05
BALD	**90.31 ± 0.32**	91.89 ± 0.17	**92.65 ± 0.13**	**92.95 ± 0.14**	93.01 ± 0.16	92.39 ± 0.08
CAL	**90.24 ± 1.27**	**92.14 ± 0.22**	**92.64 ± 0.14**	**92.92 ± 0.26**	**93.1 ± 0.28**	92.32 ± 0.12
Coreset	89.16 ± 1.37	91.9 ± 0.3	**92.63 ± 0.32**	92.64 ± 1.35	92.79 ± 0.67	**92.28 ± 0.25**
LC / BT / Entropy	**90.38 ± 0.46**	**92.3 ± 0.19**	**92.69 ± 0.21**	**93.08 ± 0.13**	**93.14 ± 0.26**	**92.49 ± 0.09**
Random	89.77 ± 0.98	90.77 ± 0.58	91.46 ± 0.26	91.67 ± 0.4	92.04 ± 0.26	91.19 ± 0.36

Table 11: Accuracy of RoBERTa on IMDB with various AL strategies with query size = 100.

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Average
BADGE	84.32 ± 0.42	**86.92 ± 0.71**	87.97 ± 0.29	**88.81 ± 1.13**	**89.34 ± 0.42**	**87.82 ± 0.36**
BALD	**84.82 ± 0.6**	**87.2 ± 0.32**	**88.67 ± 0.2**	**88.79 ± 0.51**	**89.35 ± 0.14**	**88.09 ± 0.13**
CAL	**84.13 ± 1.28**	**86.72 ± 0.55**	**88.29 ± 0.38**	**88.93 ± 0.32**	**88.86 ± 0.76**	**87.67 ± 0.46**
Coreset	84.1 ± 0.52	**86.38 ± 0.89**	87.63 ± 0.5	88.47 ± 0.42	**89.1 ± 0.31**	87.46 ± 0.33
LC / BT / Entropy	83.92 ± 1.38	**86.47 ± 1.16**	**88.41 ± 0.54**	**89.19 ± 0.32**	**89.29 ± 0.18**	**87.74 ± 0.35**
Random	82.81 ± 3.7	85.77 ± 0.61	86.84 ± 0.51	87.45 ± 0.33	88.06 ± 0.49	86.56 ± 0.5

Table 12: Accuracy of DisitlBERT on IMDB with various AL strategies with query size = 100.

Query size = 10



a) RoBERTa b) DistilBERT

Figure 8: Accuracy of the best performing query strategies with different acquisition models on IMDB with query size = 10.

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Iter. 25	Iter. 30	Average
BADGE	**69.49 ± 6.4**	**87.29 ± 1.35**	**89.16 ± 0.69**	**90.36 ± 0.45**	**90.46 ± 0.63**	**90.98 ± 0.28**	**91.13 ± 0.32**	**88.54 ± 0.53**
BALD	**67.08 ± 4.56**	84.04 ± 2.66	88.09 ± 1.61	89.49 ± 0.76	**90.56 ± 0.3**	87.85 ± 5.56	90.81 ± 0.64	87.64 ± 0.66
CAL	60.22 ± 4.41	83.73 ± 4.75	**89.33 ± 0.54**	**90.23 ± 0.26**	**90.77 ± 0.32**	**91.19 ± 0.32**	**91.4 ± 0.26**	86.81 ± 0.83
Coreset	**64.3 ± 6.06**	**86.92 ± 1.1**	88.49 ± 1.44	89.36 ± 0.79	**90.5 ± 0.32**	90.77 ± 0.25	90.72 ± 0.26	87.27 ± 1.0
LC / BT / Entropy	60.26 ± 4.86	**87.0 ± 0.87**	**89.08 ± 1.07**	**90.34 ± 0.34**	**90.35 ± 0.79**	**90.9 ± 0.49**	**91.3 ± 0.33**	**88.24 ± 0.46**
Random	**68.14 ± 5.58**	**86.6 ± 1.32**	88.53 ± 1.09	89.3 ± 0.49	89.65 ± 0.51	89.79 ± 0.81	90.01 ± 0.53	87.55 ± 0.67

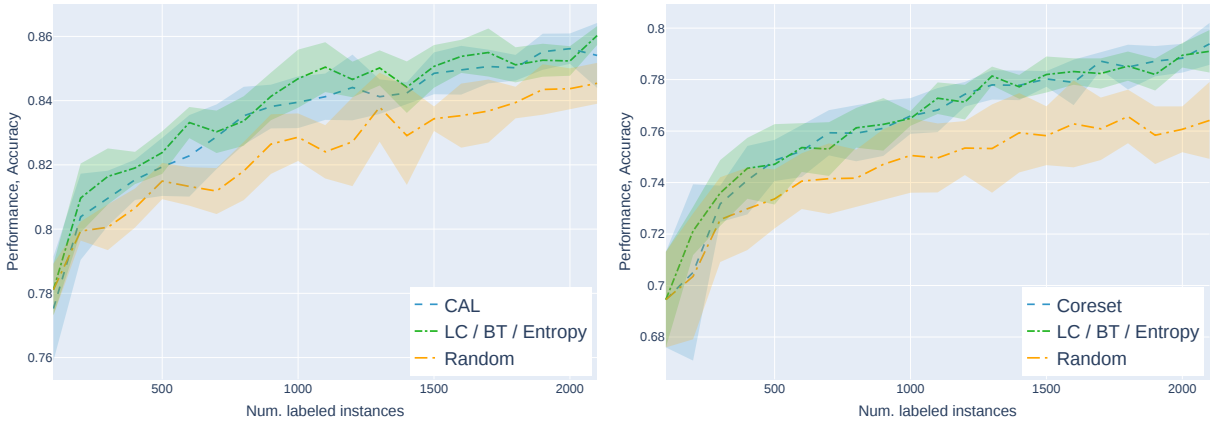
Table 13: Accuracy of RoBERTa on IMDB with various AL strategies with query size = 10.

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Iter. 25	Iter. 30	Average
BADGE	**56.53 ± 2.66**	**75.41 ± 2.65**	**81.96 ± 1.26**	**83.25 ± 1.5**	83.95 ± 1.0	84.76 ± 0.79	85.19 ± 0.93	**80.31 ± 1.21**
BALD	55.61 ± 3.27	69.51 ± 5.22	79.44 ± 3.64	83.21 ± 1.15	83.86 ± 1.34	83.08 ± 3.52	85.14 ± 1.36	78.86 ± 3.86
CAL	55.67 ± 2.69	68.58 ± 4.09	80.85 ± 1.48	83.04 ± 1.15	84.3 ± 0.78	**84.98 ± 0.47**	**85.26 ± 0.91**	78.9 ± 0.81
Coreset	54.6 ± 3.48	69.77 ± 6.89	81.33 ± 0.96	82.65 ± 1.35	83.76 ± 0.83	84.2 ± 0.64	84.7 ± 0.55	79.04 ± 1.83
LC / BT / Entropy	55.31 ± 2.62	70.53 ± 4.88	80.23 ± 2.3	83.0 ± 1.73	84.18 ± 1.11	84.21 ± 1.95	85.22 ± 0.72	79.22 ± 1.68
Random	55.45 ± 2.78	70.41 ± 5.61	80.01 ± 2.64	83.24 ± 1.6	**84.61 ± 0.48**	84.63 ± 0.68	85.21 ± 0.67	79.45 ± 1.51

Table 14: Accuracy of DistilBERT on IMDB with various AL strategies with query size = 10.

B.1.3 CoLA

Query size = 100



a) ELECTRA b) DistilBERT

Figure 9: Accuracy of the best performing query strategies with different acquisition models on CoLA with query size = 100.

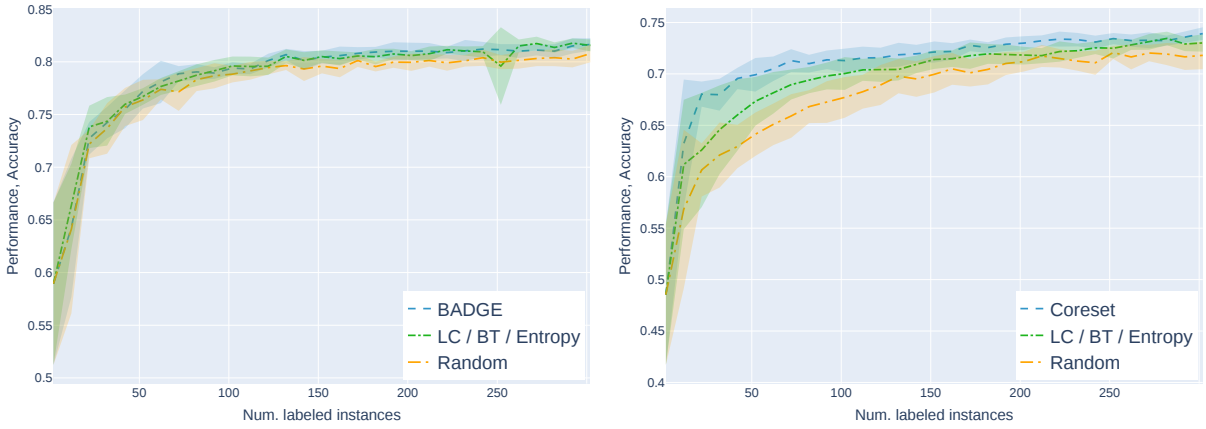
AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Average
BADGE	**80.59 ± 1.13**	**82.61 ± 0.8**	83.78 ± 0.29	**85.22 ± 0.8**	85.27 ± 0.6	83.76 ± 0.24
BALD	67.44 ± 5.74	77.34 ± 1.1	78.81 ± 0.53	79.67 ± 0.54	80.56 ± 0.4	77.73 ± 1.15
CAL	**80.38 ± 1.34**	**82.28 ± 1.27**	84.12 ± 0.72	**84.95 ± 0.76**	**85.41 ± 1.01**	**83.73 ± 0.58**
Coreset	**80.31 ± 1.14**	**82.32 ± 1.1**	84.33 ± 0.69	84.51 ± 0.62	85.16 ± 0.32	**83.57 ± 0.62**
LC / BT / Entropy	**80.97 ± 1.07**	**83.32 ± 0.49**	**85.04 ± 0.78**	**85.38 ± 0.52**	**86.03 ± 0.3**	**84.11 ± 0.24**
Random	79.94 ± 0.3	81.33 ± 0.6	82.41 ± 0.83	83.53 ± 1.0	84.54 ± 0.64	82.58 ± 0.38

Table 15: Accuracy of ELECTRA on CoLA with various AL strategies with query size = 100.

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Average
BADGE	**71.74 ± 2.1**	**75.19 ± 1.44**	**76.97 ± 0.76**	**78.09 ± 1.38**	**78.78 ± 1.17**	**76.58 ± 1.89**
BALD	59.12 ± 6.86	63.99 ± 5.21	69.3 ± 2.97	71.27 ± 1.44	72.11 ± 1.42	67.93 ± 2.05
CAL	**71.58 ± 1.34**	**75.05 ± 1.08**	**76.8 ± 0.77**	77.83 ± 0.53	**79.1 ± 0.87**	**76.58 ± 0.46**
Coreset	**70.51 ± 3.42**	**75.23 ± 1.01**	**76.82 ± 0.86**	**77.89 ± 0.88**	**79.39 ± 0.81**	**76.61 ± 0.31**
LC / BT / Entropy	**72.12 ± 0.94**	**75.36 ± 0.95**	**77.28 ± 0.61**	**78.31 ± 0.51**	**79.1 ± 0.82**	**76.71 ± 0.29**
Random	**70.35 ± 2.44**	74.06 ± 1.08	74.96 ± 1.34	76.28 ± 1.69	76.41 ± 1.49	74.8 ± 0.91

Table 16: Accuracy of DistilBERT on CoLA with various AL strategies with query size = 100.

Query size = 10



a) ELECTRA b) DistilBERT

Figure 10: Accuracy of the best performing query strategies with different acquisition models on CoLA with query size = 10.

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Iter. 25	Iter. 30	Average
BADGE	64.25 ± 6.58	**77.2 ± 1.63**	79.41 ± 0.56	80.44 ± 0.61	**81.01 ± 0.81**	81.15 ± 0.53	**81.65 ± 0.6**	**79.16 ± 1.18**
BALD	**67.41 ± 5.06**	76.7 ± 1.85	78.68 ± 0.52	79.54 ± 0.59	80.62 ± 0.47	80.7 ± 2.03	81.35 ± 0.93	78.76 ± 8.01
CAL	65.88 ± 8.86	76.59 ± 2.68	**79.7 ± 1.05**	**80.72 ± 0.43**	80.73 ± 0.67	**81.5 ± 0.49**	80.88 ± 1.33	79.05 ± 0.97
Coreset	62.89 ± 9.54	74.61 ± 3.22	79.61 ± 0.58	79.99 ± 0.96	80.57 ± 0.54	80.89 ± 0.79	81.28 ± 0.63	78.42 ± 0.82
LC / BT / Entropy	66.27 ± 4.05	76.71 ± 0.79	79.61 ± 0.9	80.5 ± 0.59	80.59 ± 0.42	79.6 ± 3.68	81.56 ± 0.57	**79.16 ± 0.43**
Random	64.11 ± 8.0	76.39 ± 1.91	78.83 ± 0.79	79.6 ± 0.69	79.95 ± 0.81	79.97 ± 0.67	80.79 ± 0.83	78.36 ± 0.6

Table 17: Accuracy of ELECTRA on CoLA with various AL strategies with query size = 10.

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Iter. 25	Iter. 30	Average
BADGE	**63.66 ± 4.45**	67.17 ± 2.53	69.64 ± 1.61	**71.74 ± 1.63**	**72.69 ± 1.06**	**73.08 ± 0.64**	**73.68 ± 0.53**	**70.63 ± 1.28**
BALD	59.39 ± 7.76	64.93 ± 2.71	68.4 ± 2.55	70.32 ± 1.18	71.2 ± 1.19	72.24 ± 1.05	**73.2 ± 0.92**	69.23 ± 1.05
CAL	60.2 ± 7.09	**69.11 ± 1.3**	**71.02 ± 1.27**	**71.82 ± 1.32**	**72.3 ± 1.37**	**72.86 ± 1.06**	**73.64 ± 0.79**	70.81 ± 0.86
Coreset	63.23 ± 6.22	**69.91 ± 1.63**	**71.29 ± 1.16**	**72.13 ± 1.03**	**72.99 ± 0.67**	**73.43 ± 0.54**	**73.92 ± 0.61**	**71.68 ± 0.67**
LC / BT / Entropy	61.19 ± 6.3	67.33 ± 2.37	**70.04 ± 1.59**	**71.4 ± 0.92**	71.77 ± 0.86	**72.49 ± 0.99**	73.01 ± 0.85	70.23 ± 1.22
Random	56.86 ± 7.72	64.13 ± 2.11	67.72 ± 1.98	69.96 ± 1.77	71.19 ± 0.96	72.08 ± 0.63	71.8 ± 1.34	68.53 ± 1.02

Table 18: Accuracy of DistilBERT on CoLA with various AL strategies with query size = 10.

B.2 Sequence Tagging

B.2.1 CoNLL-2003

Query size = 2% (tokens)

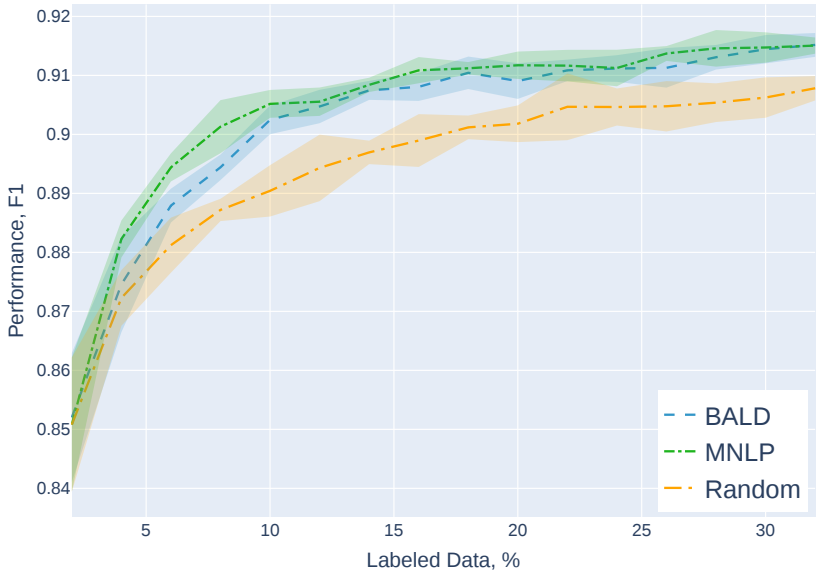
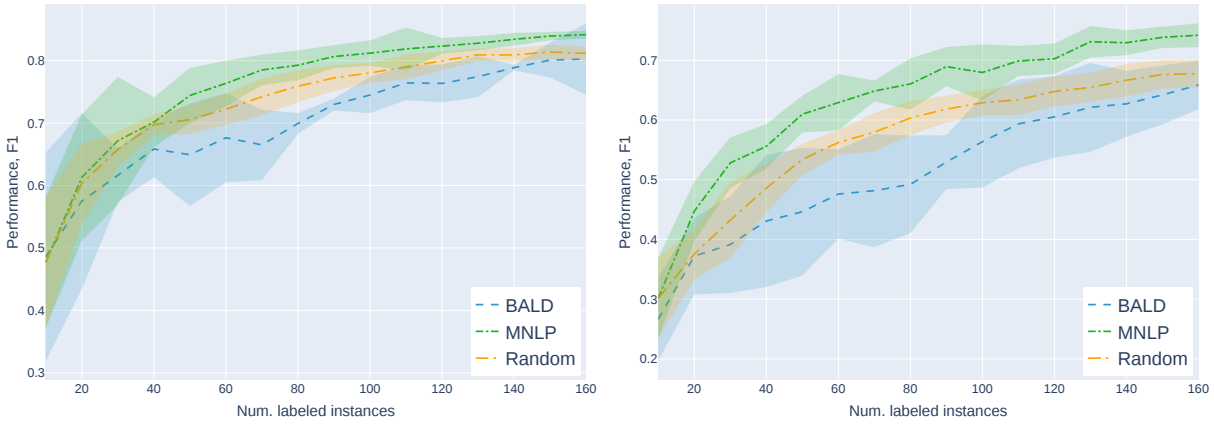


Figure 11: Overall F1-micro score of the best performing query strategies with ELECTRA on CoNLL-2003 with query size = 2% (tokens).

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Average
BALD	87.46 ± 0.75	**90.47 ± 0.28**	**91.09 ± 0.18**	**91.52 ± 0.2**	90.5 ± 0.14
MNLP	**88.23 ± 0.31**	**90.55 ± 0.24**	**91.17 ± 0.27**	**91.51 ± 0.14**	**90.75 ± 0.07**
Random	87.23 ± 0.47	89.43 ± 0.56	90.47 ± 0.57	90.78 ± 0.21	89.72 ± 0.2

Table 19: Overall F1-micro score of ELECTRA on CoNLL-2003 with various AL strategies with query size = 2% (tokens).

Query size = 100 (tokens)



a) ELECTRA b) DistilBERT

Figure 12: Overall F1-micro score of the best performing query strategies with different acquisition models on CoNLL-2003 with query size = 100.

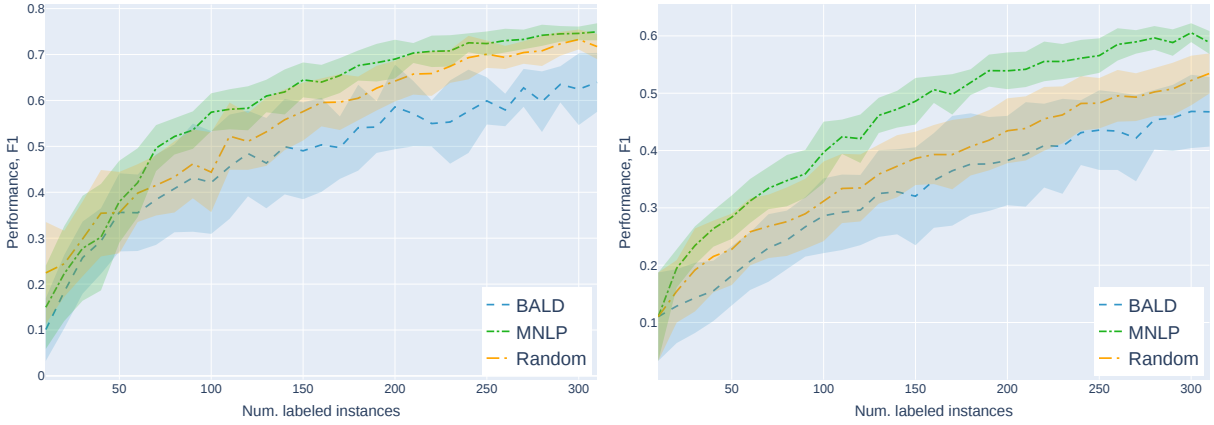
AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Average
BALD	57.52 ± 14.08	67.65 ± 7.13	76.44 ± 2.76	80.27 ± 5.72	71.4 ± 2.21
MNLP	**61.32 ± 10.1**	**76.36 ± 3.65**	**81.87 ± 3.44**	**84.16 ± 0.61**	**77.84 ± 2.59**
Random	60.35 ± 6.36	72.22 ± 2.52	78.99 ± 1.98	81.16 ± 1.07	75.16 ± 1.57

Table 20: Overall F1-micro score of ELECTRA on CoNLL-2003 with various AL strategies with query size = 100 (tokens).

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Average
BALD	37.21 ± 6.39	47.6 ± 7.48	59.34 ± 7.41	65.86 ± 4.07	52.88 ± 9.22
MNLP	**44.73 ± 5.0**	**62.96 ± 4.71**	**69.88 ± 2.53**	**74.2 ± 2.0**	**65.27 ± 1.64**
Random	37.53 ± 4.21	56.22 ± 2.09	63.39 ± 2.57	67.76 ± 2.29	58.51 ± 1.81

Table 21: Overall F1-micro score of DistilBERT on CoNLL-2003 with various AL strategies with query size = 100 (tokens).

Query size = 10 (tokens)



a) ELECTRA b) DistilBERT

Figure 13: Overall F1-micro score of the best performing query strategies with different acquisition models on CoNLL-2003 with query size = 10.

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Iter. 25	Iter. 30	Average
BALD	18.34 ± 7.99	35.53 ± 8.32	45.57 ± 11.36	50.33 ± 10.24	57.12 ± 7.12	57.93 ± 3.54	63.94 ± 6.42	49.02 ± 15.28
MNLP	22.28 ± 10.34	**42.1 ± 7.49**	**58.07 ± 4.37**	**63.99 ± 3.77**	**70.33 ± 2.17**	**73.03 ± 2.58**	**74.95 ± 1.86**	**60.41 ± 3.31**
Random	**24.43 ± 7.2**	39.82 ± 6.3	**52.22 ± 7.31**	**59.54 ± 5.19**	65.74 ± 4.47	69.36 ± 2.52	71.74 ± 2.75	56.11 ± 3.99

Table 22: Overall F1-micro score of ELECTRA on CoNLL-2003 with various AL strategies with query size = 10 (tokens).

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Iter. 20	Iter. 25	Iter. 30	Average
BALD	12.85 ± 6.45	20.7 ± 5.03	29.23 ± 6.59	34.78 ± 8.26	39.33 ± 9.11	43.39 ± 6.78	46.76 ± 6.07	33.43 ± 6.15
MNLP	**19.46 ± 3.38**	**31.2 ± 3.85**	**42.4 ± 3.0**	**50.63 ± 2.33**	**54.17 ± 3.1**	**58.52 ± 2.82**	**58.85 ± 2.0**	**46.42 ± 2.31**
Random	**15.43 ± 5.45**	25.82 ± 5.65	33.39 ± 6.04	39.3 ± 5.16	43.86 ± 5.6	49.55 ± 4.51	53.45 ± 3.49	38.0 ± 4.56

Table 23: Overall F1-micro score of DistilBERT on CoNLL-2003 with various AL strategies with query size = 10 (tokens).

B.3 Abstractive Text Summarization

B.3.1 AESLC

Query size = 10

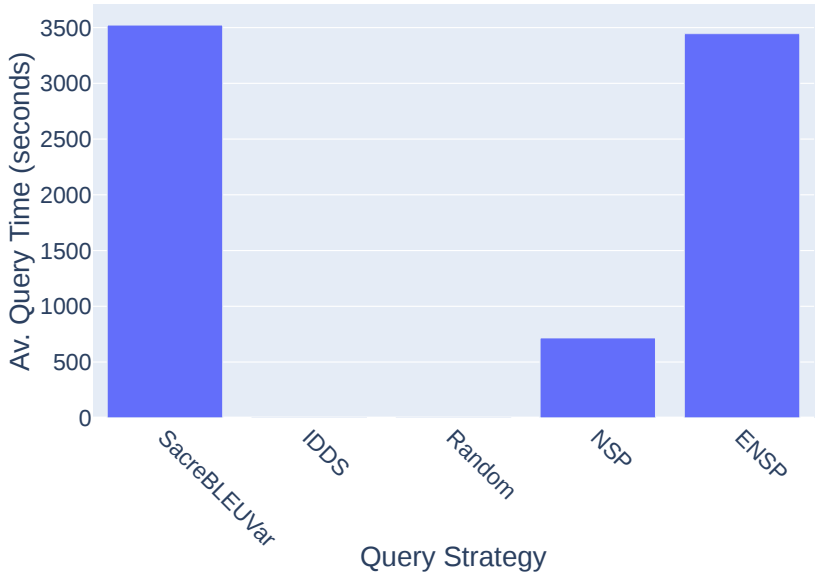
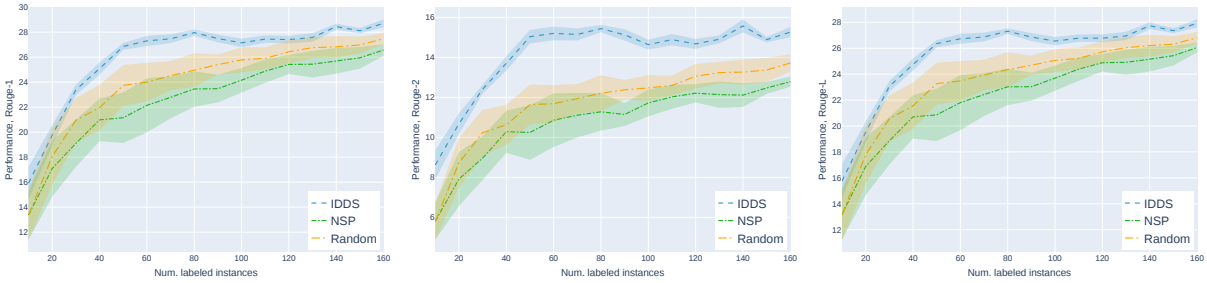
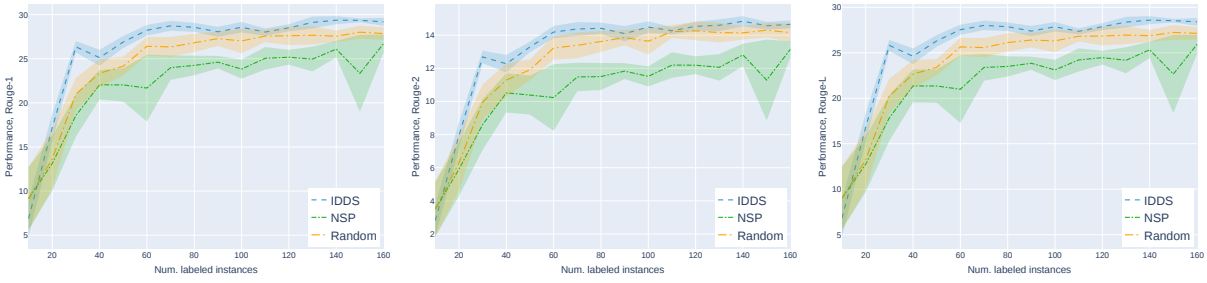


Figure 14: Average duration in seconds of one AL query with different strategies on AESLC with BART as an acquisition model and query size = 10. Hardware configuration is provided in Appendix C.



a) ROUGE-1 a) ROUGE-2 a) ROUGE-L

Figure 15: ROUGE scores of the best performing query strategies with BART as an acquisition model on AESLC with query size = 10.



a) ROUGE-1 a) ROUGE-2 a) ROUGE-L

Figure 16: ROUGE scores of the best performing query strategies with PEGASUS as an acquisition model on AESLC with query size = 10.

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Average
ROUGE-1					
SacreBLEUVar	**18.77 ± 2.21**	23.46 ± 1.18	26.25 ± 0.69	27.53 ± 0.45	24.6 ± 0.96
IDDS	**19.76 ± 0.82**	**27.3 ± 0.41**	**27.44 ± 0.3**	**28.7 ± 0.32**	**26.67 ± 0.08**
Random	**18.07 ± 2.26**	24.0 ± 1.55	25.9 ± 0.88	27.47 ± 0.45	24.65 ± 0.94
NSP	17.09 ± 2.25	22.15 ± 2.16	24.88 ± 0.94	26.56 ± 0.46	23.22 ± 0.97
ENSP	14.57 ± 2.92	23.2 ± 1.16	25.41 ± 0.79	27.42 ± 0.92	23.74 ± 1.24
ROUGE-2					
SacreBLEUVar	9.0 ± 1.12	11.26 ± 0.89	12.86 ± 0.62	13.59 ± 0.56	12.05 ± 0.71
IDDS	**10.69 ± 0.53**	**15.2 ± 0.34**	**14.89 ± 0.29**	**15.26 ± 0.26**	**14.51 ± 0.07**
Random	8.73 ± 1.24	11.69 ± 0.93	12.6 ± 0.49	13.72 ± 0.43	12.08 ± 0.63
NSP	7.92 ± 1.35	10.86 ± 1.34	12.03 ± 0.59	12.79 ± 0.26	11.15 ± 0.66
ENSP	6.69 ± 1.51	11.37 ± 0.67	12.36 ± 0.46	13.5 ± 0.58	11.52 ± 0.66
ROUGE-L					
SacreBLEUVar	**18.51 ± 2.19**	22.93 ± 1.17	25.64 ± 0.73	26.91 ± 0.51	24.08 ± 0.99
IDDS	**19.52 ± 0.81**	**26.73 ± 0.39**	**26.78 ± 0.29**	**27.92 ± 0.31**	**26.1 ± 0.08**
Random	**17.79 ± 2.21**	23.48 ± 1.53	25.22 ± 0.84	26.81 ± 0.44	24.07 ± 0.92
NSP	16.88 ± 2.21	21.81 ± 2.13	24.39 ± 0.9	26.05 ± 0.38	22.81 ± 0.95
ENSP	14.41 ± 2.89	22.76 ± 1.12	24.93 ± 0.75	26.85 ± 0.92	23.3 ± 1.2

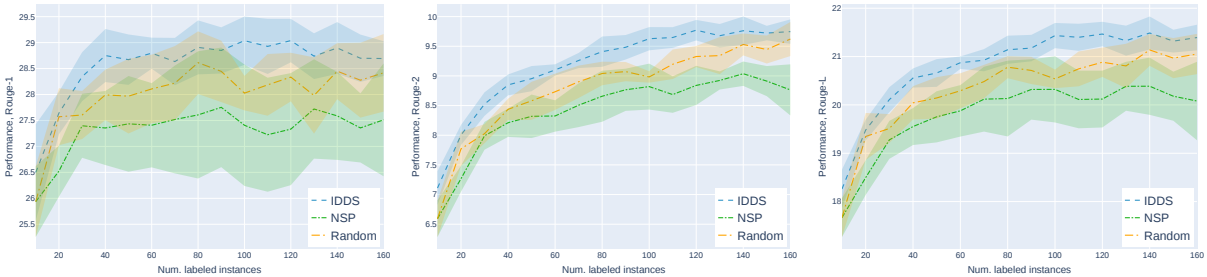
Table 24: ROUGE scores of BART on AESLC with various AL strategies with query size = 10.

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Average
ROUGE-1					
SacreBLEUVar	15.5 ± 1.59	25.11 ± 1.05	26.47 ± 1.19	28.02 ± 0.83	25.13 ± 0.58
IDDS	**17.19 ± 1.58**	**28.23 ± 0.61**	**28.02 ± 0.42**	**29.18 ± 0.45**	**27.42 ± 0.15**
Random	13.65 ± 3.34	26.42 ± 1.05	**27.57 ± 0.7**	27.86 ± 0.71	25.49 ± 0.8
NSP	13.12 ± 3.11	21.68 ± 3.79	25.05 ± 1.28	26.72 ± 1.02	23.03 ± 0.89
ENSP	12.39 ± 2.14	22.27 ± 2.13	23.37 ± 4.21	26.87 ± 0.55	23.36 ± 1.18
ROUGE-2					
SacreBLEUVar	7.01 ± 0.86	12.67 ± 0.63	13.67 ± 0.54	**14.72 ± 0.6**	12.74 ± 0.49
IDDS	**7.94 ± 0.81**	**14.19 ± 0.37**	**14.27 ± 0.22**	**14.65 ± 0.24**	**13.68 ± 0.11**
Random	**6.3 ± 1.68**	13.23 ± 0.71	**14.21 ± 0.39**	**14.15 ± 0.56**	12.83 ± 0.49
NSP	5.91 ± 1.62	10.24 ± 2.01	12.2 ± 0.77	13.14 ± 0.49	11.04 ± 0.56
ENSP	5.25 ± 0.9	10.53 ± 1.22	11.23 ± 2.42	12.9 ± 0.82	11.15 ± 0.69
ROUGE-L					
SacreBLEUVar	15.07 ± 1.55	24.47 ± 0.84	25.81 ± 0.75	27.2 ± 0.83	24.43 ± 0.57
IDDS	**16.8 ± 1.57**	**27.53 ± 0.56**	**27.36 ± 0.33**	**28.4 ± 0.43**	**26.75 ± 0.15**
Random	13.24 ± 3.23	25.65 ± 0.99	**26.82 ± 0.58**	27.13 ± 0.69	24.76 ± 0.75
NSP	12.75 ± 3.07	20.99 ± 3.72	24.21 ± 1.27	25.93 ± 1.04	22.33 ± 0.93
ENSP	12.0 ± 2.11	21.63 ± 2.12	22.71 ± 4.2	26.12 ± 0.61	22.71 ± 1.21

Table 25: ROUGE scores of PEGASUS on AESLC with various AL strategies with query size = 10.

B.3.2 WikiHow

Query size = 10



a) ROUGE-1 a) ROUGE-2 a) ROUGE-L

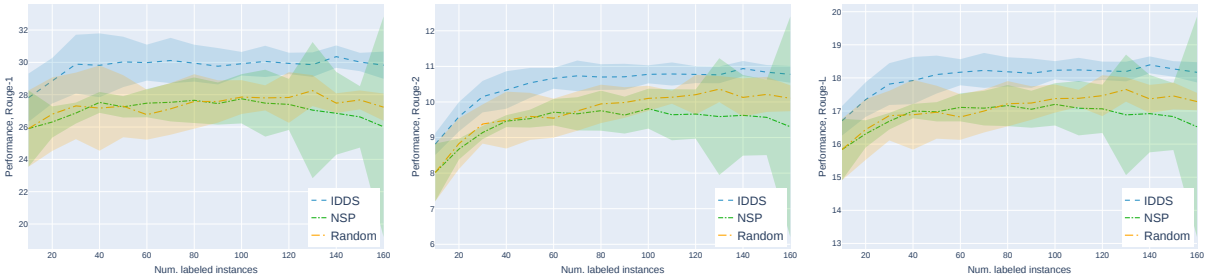
Figure 17: ROUGE scores of the best performing query strategies with BART as an acquisition model on WikiHow with query size = 10.

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Average
			ROUGE-1		
BLEUVar	26.88 ± 0.6	27.61 ± 0.72	27.64 ± 0.64	28.08 ± 0.39	27.65 ± 0.41
IDDS	**27.64 ± 0.41**	**28.8 ± 0.3**	**28.93 ± 0.53**	**28.69 ± 0.33**	**28.71 ± 0.36**
Random	**27.57 ± 0.55**	**28.11 ± 0.66**	28.19 ± 0.59	**28.42 ± 0.75**	28.15 ± 0.49
NSP	26.52 ± 0.49	27.41 ± 0.81	27.23 ± 1.1	27.51 ± 1.09	27.41 ± 0.85
ENSP	26.91 ± 0.47	27.6 ± 0.48	27.53 ± 0.8	27.83 ± 0.93	27.52 ± 0.71
			ROUGE-2		
BLEUVar	7.43 ± 0.3	8.64 ± 0.22	9.19 ± 0.2	**9.54 ± 0.23**	8.83 ± 0.16
IDDS	**8.0 ± 0.17**	**9.1 ± 0.1**	**9.65 ± 0.17**	**9.75 ± 0.21**	**9.3 ± 0.13**
Random	7.77 ± 0.21	8.73 ± 0.31	9.2 ± 0.25	**9.62 ± 0.29**	8.93 ± 0.13
NSP	7.27 ± 0.22	8.32 ± 0.26	8.69 ± 0.31	8.76 ± 0.43	8.54 ± 0.35
ENSP	7.4 ± 0.18	8.43 ± 0.17	8.61 ± 0.34	8.85 ± 0.4	8.44 ± 0.26
			ROUGE-L		
BLEUVar	18.82 ± 0.37	20.07 ± 0.41	20.56 ± 0.34	20.9 ± 0.36	20.25 ± 0.21
IDDS	**19.48 ± 0.21**	**20.87 ± 0.13**	**21.4 ± 0.28**	**21.39 ± 0.27**	**20.98 ± 0.18**
Random	**19.35 ± 0.49**	20.29 ± 0.36	20.75 ± 0.36	**21.05 ± 0.42**	20.5 ± 0.23
NSP	18.5 ± 0.36	19.88 ± 0.53	20.11 ± 0.6	20.08 ± 0.81	19.94 ± 0.46
ENSP	18.65 ± 0.27	19.67 ± 0.29	20.43 ± 0.72	20.71 ± 0.43	19.97 ± 0.6

Table 26: ROUGE scores of BART on WikiHow with various AL strategies with query size = 10.

B.3.3 PubMed

Query size = 10



a) ROUGE-1 a) ROUGE-2 a) ROUGE-L

Figure 18: ROUGE scores of the best performing query strategies with BART as an acquisition model on PubMed with query size = 10.

AL Strategy	Iter. 1	Iter. 5	Iter. 10	Iter. 15	Average
ROUGE-1					
BLEUVar	26.07 ± 1.01	27.92 ± 1.2	27.91 ± 1.0	27.52 ± 0.93	27.51 ± 0.87
IDDS	**28.86 ± 1.41**	**29.98 ± 1.11**	**30.06 ± 0.97**	**29.83 ± 0.84**	**29.89 ± 0.97**
Random	26.8 ± 2.27	26.77 ± 1.55	27.8 ± 0.78	27.23 ± 0.84	27.45 ± 1.37
NSP	26.31 ± 0.97	27.48 ± 0.87	27.48 ± 2.07	26.02 ± 6.82	27.15 ± 1.18
ENSP	26.16 ± 1.16	27.75 ± 1.56	27.65 ± 1.32	27.53 ± 1.62	27.51 ± 1.12
ROUGE-2					
BLEUVar	8.61 ± 0.27	9.95 ± 0.31	10.26 ± 0.23	10.17 ± 0.25	9.92 ± 0.22
IDDS	**9.58 ± 0.41**	**10.66 ± 0.29**	**10.79 ± 0.32**	**10.78 ± 0.24**	**10.6 ± 0.27**
Random	8.83 ± 0.71	9.55 ± 0.55	10.13 ± 0.18	10.1 ± 0.36	9.85 ± 0.39
NSP	8.68 ± 0.29	9.71 ± 0.37	9.64 ± 0.72	9.3 ± 3.1	9.52 ± 0.44
ENSP	8.66 ± 0.35	9.87 ± 0.47	10.21 ± 0.35	10.18 ± 0.41	9.89 ± 0.32
ROUGE-L					
BLEUVar	16.19 ± 0.37	17.37 ± 0.4	17.53 ± 0.31	17.41 ± 0.34	17.26 ± 0.29
IDDS	**17.34 ± 0.52**	**18.17 ± 0.39**	**18.24 ± 0.37**	**18.17 ± 0.31**	**18.11 ± 0.35**
Random	16.44 ± 0.89	16.82 ± 0.69	17.38 ± 0.26	17.29 ± 0.26	17.16 ± 0.51
NSP	16.31 ± 0.41	17.11 ± 0.42	17.08 ± 0.82	16.52 ± 3.33	16.93 ± 0.54
ENSP	16.23 ± 0.46	17.27 ± 0.57	17.46 ± 0.45	17.39 ± 0.54	17.23 ± 0.38

Table 27: ROUGE scores of BART on PubMed with various AL strategies with query size = 10.

C Computationally Efficient AL

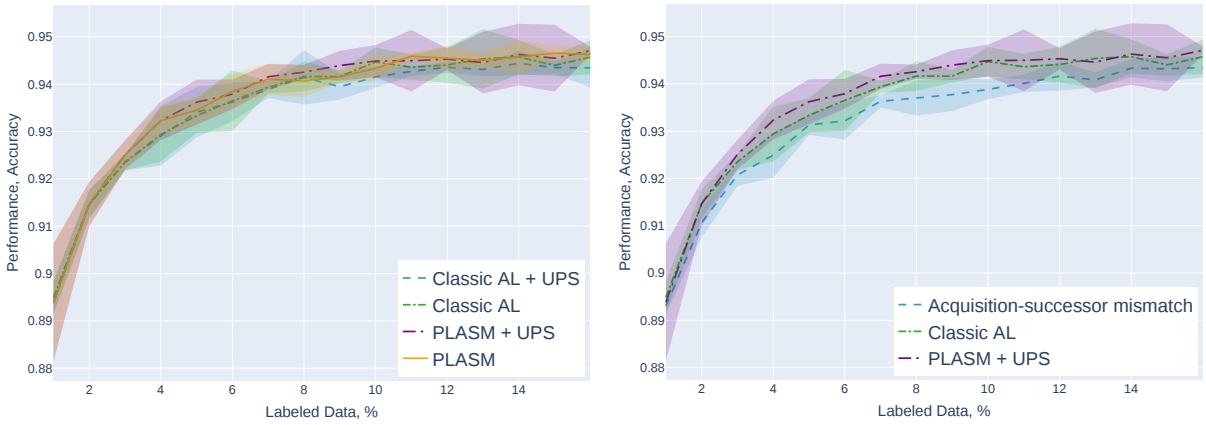
Hardware Configuration

We use the following hardware configuration for the experiments: 2 Intel Xeon Platinum 8168, 2.7 GHz, 24 cores CPU; NVIDIA Tesla v100 GPU, 32 Gb of VRAM. The results are averaged across 5 runs with different seeds to ensure stability.

Experiment Hyperparameters

On each iteration, we select 1% of instances according to AL strategy for text classification datasets, and 2% of instances for sequence tagging. For UPS, we use $\gamma = 0.1, T = 0.01$, and recalculate the uncertainty estimates for the whole dataset on the 0-th, 1-st, 4-th, and 8-th iterations.

C.1 AG News

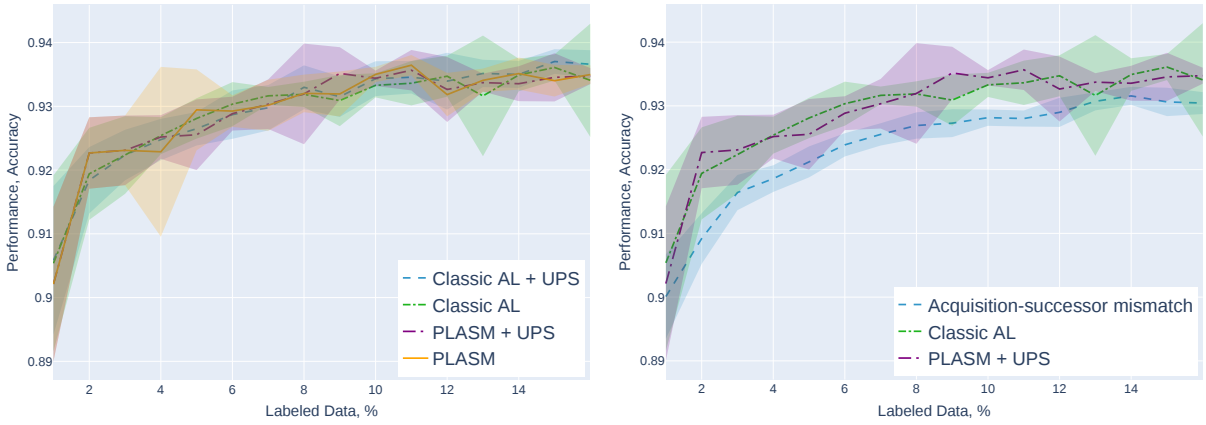


a) PLASM and UPS v Classic AL

b) ASM problem

Figure 19: AG News dataset: performance of PLASM and UPS algorithms compared to classic AL and acquisition-successor mismatch (ASM) settings. For all the experiments, ELECTRA is used as a successor model (therefore, as an acquisition model in "classic AL" as well), and DistilBERT – for acquisition in PLASM and ASM.

C.2 IMDB



a) PLASM and UPS v Classic AL

b) ASM problem

Figure 20: IMDB dataset: performance of PLASM and UPS algorithms compared to classic AL and acquisition-successor mismatch (ASM) settings. For all the experiments, RoBERTa is used as a successor model (therefore, as an acquisition model in "classic AL" as well), and DistilELECTRA – for acquisition in PLASM and ASM.

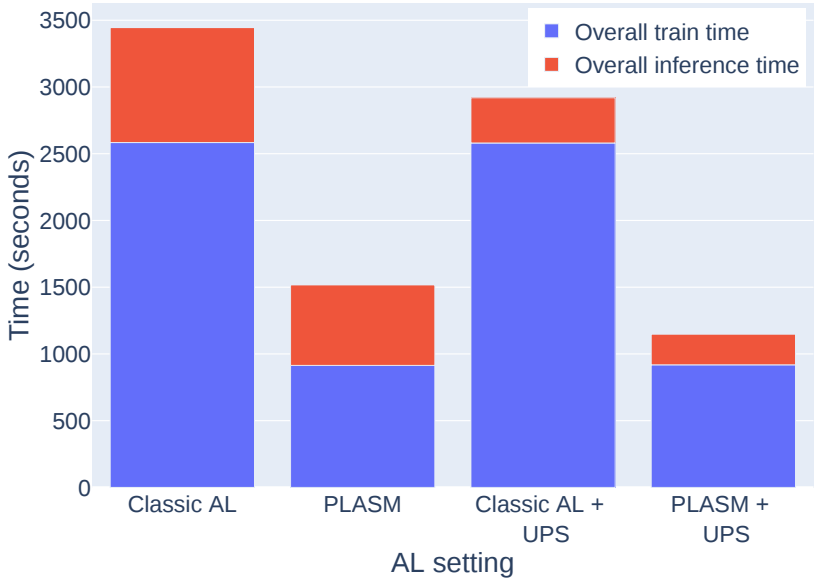
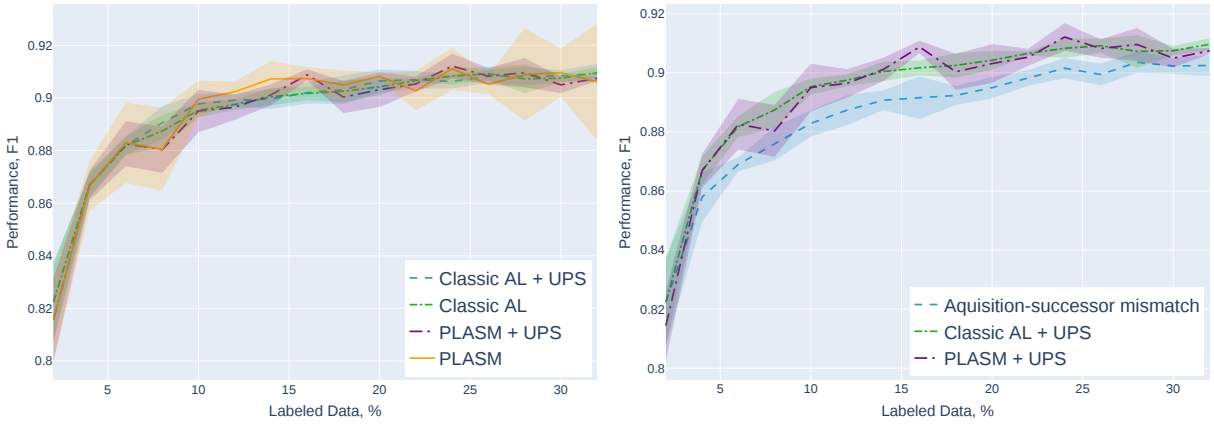


Figure 21: Duration in seconds of all the training and inference phases of the simulated AL with different acquisition settings on IMDB with query size = 1% and 15 AL iterations. RoBERTa is used as a successor model, and DistilELECTRA – for acquisition in PLASM.

C.3 CoNLL-2003



a) PLASM and UPS v Classic AL

b) ASM problem

Figure 22: CoNLL dataset: performance of PLASM and UPS algorithms compared to classic AL and acquisition-successor mismatch (ASM) settings. For all the experiments, ELECTRA is used as a successor model (therefore, as an acquisition model in "classic AL" as well), and DistilBERT – for acquisition in PLASM and ASM.

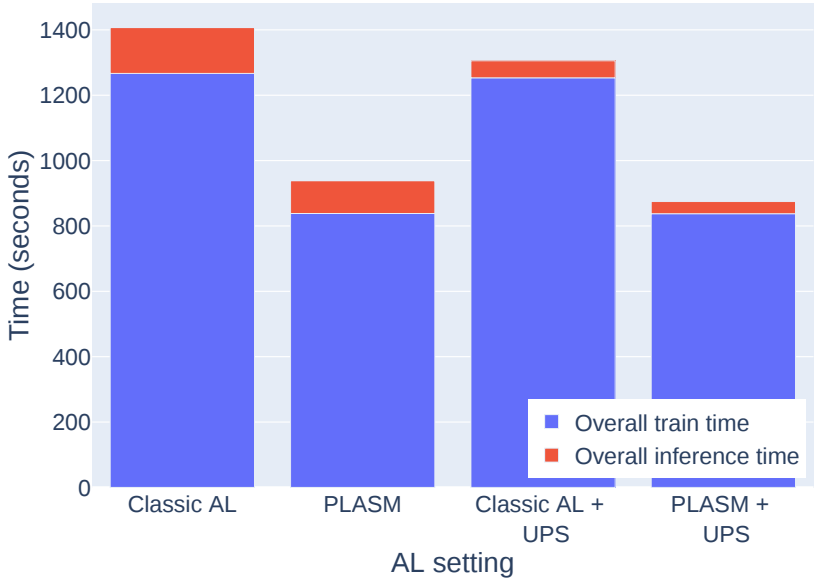


Figure 23: Duration in seconds of all the training and inference phases of the simulated AL with different acquisition settings on CoNLL-2003 with query size = 2% and 15 AL iterations. ELECTRA is used as a successor model, and DistilBERT – for acquisition in PLASM.