Towards Computationally Feasible Deep Active Learning

Anonymous ACL submission

Abstract

Active learning (AL) is a prominent technique for reducing the annotation effort required for training machine learning models. Deep learning offers a solution for several essential obstacles to deploying AL in practice but introduces many others. One of such problems is the excessive computational resources required to train an acquisition model and estimate its uncertainty on instances in the unlabeled pool. We propose two techniques that tackle this issue for text classification and tagging tasks, offering a substantial reduction of AL iteration duration and the computational overhead introduced by deep acquisition models in AL. We also demonstrate that our algorithm that leverages pseudolabeling and distilled models overcomes one of the essential obstacles revealed previously in the literature. Namely, it was shown that due to differences between an acquisition model used to select instances during AL and a successor model trained on the labeled data, the benefits of AL can diminish. We show that our algorithm, despite using a smaller and faster acquisition model, is capable of training a more expressive successor model with higher performance.

1 Introduction

Active learning (AL) (Cohn et al., 1996) is an approach for reducing the amount of dataset annotation required for achieving the desired level of machine learning model performance. This is especially important in domains where obtaining labeled instances is expensive or wide crowdsourcing is unavailable. For example, annotation of clinical and biomedical texts usually requires the help of physicians or biomedical researchers. The time of such highly qualified experts is extremely valuable and should be spent wisely. Straightforward annotation of datasets can be very redundant, wasting the time of annotators on unimportant instances. AL alleviates this problem by asking human experts to label only the most informative instances selected according to the information acquired from a machine learning model. The algorithm for selection of such instances is called a *query strategy*, and a model used to estimate the informativeness of yet unlabeled instances is called an *acquisition model*.

AL starts from a small *seeding set* of labeled instances, which are used to train an initial acquisition model. A query strategy ranks unlabeled instances in a large pool according to a criterion that measures their informativeness based on the acquisition model output. One of the most widely adopted criteria is the uncertainty of the acquisition model on instances in question (Lewis and Gale, 1994). Eventually, top selected instances are presented to annotators, and this active annotation process iteratively continues.

After labels are collected, we would like to train a model for a final application. In the same vein as (Lowell et al., 2019), we call it a *successor model*. AL can help reduce the amount of annotation required to achieve a reasonable quality of the successor text processing model by multiple times (Settles and Craven, 2008; Settles, 2009).

Recently, deep learning has given us a tool for solving one of the essential problems of AL. When we start an annotation, we have to build an acquisition model almost without insights from the data that could help us to perform feature engineering or designing inductive bias. Deep learning does not require feature engineering, and transfer learning with deep pre-trained models like ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), and their successors such as ELECTRA (Clark et al., 2020) provide near state-of-the-art performance on a variety of tasks without any modifications to their architectures. However, deep learning introduces another problem related to computational performance. Since AL annotation typically is an interactive process, we have to train acquisition models and perform inference on a huge unlabeled pool of instances very quickly. This imposes constraints on the acquisition model size and entails another issue.

Ideally, the architectures of acquisition and successor models should be the same. Lowell et al. (2019) demonstrate that when the acquisition model is different from the successor model, the performance of the latter can degrade compared to the performance of the model trained on the same amount of annotation obtained without AL. The performance drop in the case of acquisition-successor mismatch raises the question of whether AL is a practical technique at all since this is a common situation to try various models on the annotated dataset. It also leads to a contradiction between the fact that we would like the acquisition model to be as lightweight as possible to mitigate computational overhead and the successor model to be as expressive as possible because we apparently care about the quality of our final application.

In this work, we propose a simple algorithm based on pseudo-labeling and demonstrate that it is able to alleviate the acquisition-successor mismatch problem. Moreover, we show that it is possible to substitute a resource-intensive acquisition model with a smaller one (e.g., take DistilBERT instead of BERT) but train a more powerful successor model of an arbitrary type (e.g., ELECTRA) without loss of quality. This helps to accelerate the execution of AL iterations and reduce computational overhead.

We also find that the most time-consuming part of an AL iteration with uncertainty-based query strategies can be the inference on the unlabeled pool of instances, while a set of the most certain instances usually does not change substantially from iteration to iteration. Therefore, the straightforward approach to instance acquisition wastes much time on instances shown to be unimportant in previous iterations. We leverage this finding and propose an algorithm that subsamples instances in the unlabeled pool depending on their uncertainty scores obtained on previous AL iterations. This helps to speed up the AL iterations further, especially when the unlabeled pool is large. A series of experiments on text classification and tagging benchmarks widely used in recent works on AL demonstrate the efficiency of the proposed algorithms.

The contributions of the paper are the following:

- We propose a novel algorithm denoted as **P**seudo-**L**abeling for **A**cquisition **S**uccessor **M**ismatch (PLASM) that allows the use of computationally cheap models during the acquisition of instances in AL, while it does not introduce constraints on the type of the successor model and effectively alleviates the acquisition-successor mismatch problem. It helps to reduce the hardware requirements and the duration of AL iterations.
- We propose a novel algorithm denoted as **U**nlabeled **P**ool **S**ubsampling (UPS) that helps to reduce the time required for calculating informativeness of instances in AL based on the fact that the set of instances that model is certain about does not change substantially. This helps to further speed up the AL iteration.

2 Related Work

Deep learning, to a large extent, has freed data scientists from doing feature engineering, which has been one of the essential obstacles to annotation with AL. This advantage has sparked a series of works on deep active learning (DAL) in natural language processing (NLP) that investigate the combination of these two techniques.

Shen et al. (2017) conduct one of the first investigations on DAL in sequence tagging tasks. They propose an efficient way of quantifying the uncertainty of sentences, namely maximal normalized log probability (MNLP), by averaging log probabilities of their tokens. They also address the problem of excessive duration of a neural network training step during an AL iteration by interleaving online learning with training from scratch. In our work, we take MNLP as a query strategy for experiments on sequence tagging tasks since it has demonstrated a good trade-off between quality and computational performance. We consider that online learning can potentially be used as a complement to our algorithms. Since the most time-consuming part of an AL iteration can be model inference instead of training, in this work, we also pay attention to the acceleration of the inference step.

Several recent publications investigate deep pre-trained models based on the Transformer architecture (Vaswani et al., 2017), ELMo (Peters et al., 2018), and ULMFiT (Howard and Ruder, 2018) in AL on NLP tasks (Prabhu et al., 2019; Ein-Dor et al., 2020; Yuan et al., 2020; Shelmanov et al., 2021). We continue this line of works by relying on pre-trained Transformers since this architecture has been shown promising for AL in NLP due to its good qualitative and computational performance.

Few works have experimented with Bayesian query strategies for AL. Shen et al. (2017), Siddhant and Lipton (2018), Ein-Dor et al. (2020), and Shelmanov et al. (2021) leverage Monte Carlo dropout (Gal and Ghahramani, 2016) for quantifying uncertainty of models. Siddhant and Lipton (2018) also apply the Bayes by backprop algorithm (Blundell et al., 2015) for performing variational inference of a Bayesian neural network. This approach demonstrates the best improvements upon the baseline but introduces large computational overhead both for training and uncertainty estimation of a model, as well as the memory overhead for storing parameters of a Bayesian neural network. The query strategies based on Monte Carlo dropout do not affect the model training procedure and do not change the memory footprint. However, they also suffer from slow uncertainty estimation due to necessity of making multiple stochastic predictions, while their empirical evaluations with Transformers in recent works (Ein-Dor et al., 2020; Shelmanov et al., 2021) do not demonstrate significant advantages. Therefore, we do not use Bayesian query strategies in our experiments and adhere to the classical uncertainty-based query strategies.

Recently proposed alternatives to uncertainty-based query strategies leverage reinforcement learning and imitation learning (Fang et al., 2017; Liu et al., 2018; Vu et al., 2019; Brantley et al., 2020). This series of works aims at constructing trainable policy-based query strategies. Learning such policies is a challenging task, requiring an enormous amount of computation for obtaining a supervision signal, especially when an acquisition model is a deep neural network. Such an approach can be practical only when a policy is pre-trained beforehand the actual annotation process as suggested in (Fang et al., 2017; Liu et al., 2018). However, the transferability of learned policies across domains and tasks is currently underexplored.

Finally, Lowell et al. (2019) question the usefulness of AL techniques in general. They demonstrate that due to the acquisition-successor mismatch problem, AL can be even detrimental to the performance of the successor. This finding is also revealed for classical machine learning models by Baldridge and Osborne (2004), Tomanek and Morik (2011), Hu et al. (2016) and supported by experiments with Transformers in (Shelmanov et al., 2021). Our work directly addresses the question raised by Lowell et al. (2019) and suggests a simple solution to the acquisition-successor mismatch problem. Moreover, we combine it with the method proposed by Shelmanov et al. (2021), who suggest using distilled models for instance acquisition and their teacher models as successors.

3 Background and Methods

This section describes models and AL query strategies used in the experiments and outlines the proposed algorithms.

3.1 Query Strategies

We conduct experiments with three basic AL query strategies. We note that despite their simplicity, these strategies are usually on par with more elaborated counterparts (Ein-Dor et al., 2020; Shelmanov et al., 2021; Margatina et al., 2021).

Random sampling is used for both text classification and sequence tagging experiments. Applying this strategy means that we do not use AL at all and just emulate that an annotator labels a randomly sampled piece of a dataset.

Least Confident (LC) is used for text classification experiments. This strategy sorts texts in the ascending order of their maximum class probabilities given by a machine learning model. Let y be a predicted class of an instance x, then LC_{cls} is:

$$LC_{cls} = 1 - \max_{y} \mathbb{P}\left(y|x\right).$$

Maximum Normalized Log-Probability (MNLP) is proposed by Shen et al. (2017) to mitigate the drawback of the standard LC when it is applied to sequence tagging tasks. Let y_i be a tag of a token i, let x_j be a token j in an input sequence of length n. Then the MNLP score can be formulated as follows:

$$MNLP = - \max_{y_1,\ldots,y_n} \frac{1}{n} \sum_{i}^{n} \log \mathbb{P}\left[y_i|\{y_j\} \setminus y_i, \{\mathbf{x}_j\}\right]$$

This modified version of LC works slightly better for sequence tagging tasks (Shen et al., 2017), and is adopted in many other works on DAL (Siddhant and Lipton, 2018; Erdmann et al., 2019; Shelmanov et al., 2021).

3.2 Models

We use the standard models based on the Transformer architecture (Vaswani et al., 2017) proposed by Devlin et al. (2019) and Clark et al. (2020). For

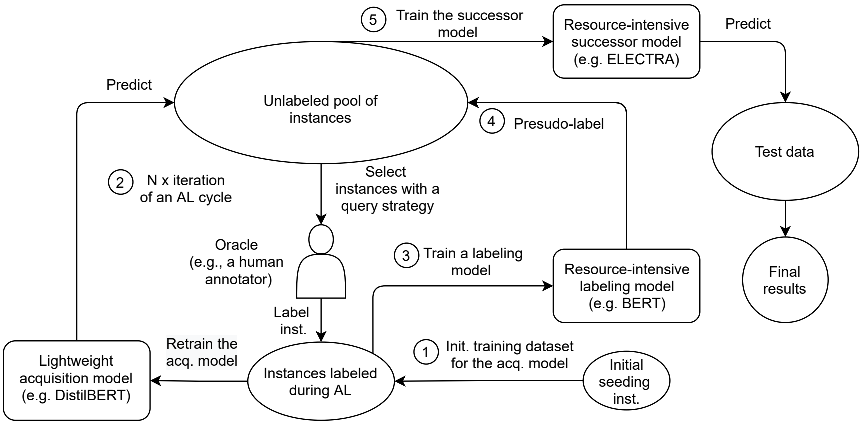


Figure 1: The scheme of the Pseudo-labeling for Acquisition-Successor Mismatch algorithm

sequence tagging, the models consist of a Transformer encoder and a classification head. For text classification, they also include a pooling mechanism. We also employ a CNN-BiLSTM-CRF model of Ma and Hovy (2016) for ablation study.

Besides full-fledged Transformers, we leverage the distilled version of BERT: DistilBERT (Sanh et al., 2019). The distillation procedure aims at creating a smaller-size model (student) while keeping the behavior of the original model (teacher) by minimizing the distillation loss over the student predictions and soft target probabilities of the teacher (Hinton et al., 2015):

$$L_{distil} = -\sum_{i,c} t_{ic} * \log\left(s_{ic}\right)$$

where t_{ic} and s_{ic} are probabilities estimated by the teacher and the student correspondingly for each instance i and class c. DistilBERT also takes advantage of several additional techniques that help align it with BERT.

DistilBERT is much more compact than its teacher. It contains 66M of parameters compared to 110M in BERT-base, which results in a 40% reduction of a memory footprint. The distilled model also achieves the 60% speedup, sacrificing only 3% of its qualitative performance (Sanh et al., 2019). Since the qualitative performance during acquisition is not essential, we would like to use such lightweight models for instance acquisition to reduce AL iteration duration and the requirements to the computational power of the hardware.

3.3 Pseudo-labeling for Acquisition-Successor Mismatch

We propose a simple algorithm for constructing a successor model of an arbitrary type using AL: *Pseudo-Labeling for Acquisition-Successor Mismatch (PLASM)*. The algorithm is designed for reducing the amount of computation required for instance acquisition during AL with uncertainty-based query strategies.

PLASM leverages the finding of Shelmanov et al. (2021) that the successor model can be trained on instances labeled during AL without a penalty to the quality if its distilled version was used for instance acquisition. However, this idea alone does not resolve the question, how we can train new models of arbitrary type on datasets collected via AL (Lowell et al., 2019).

The algorithm scheme is presented in Figure 1:

1. Consider we have a resource-intensive pre-trained teacher model (e.g. BERT). We construct a lightweight distilled version of this model (e.g. DistilBERT) using unlabeled data.
2. We apply a distilled model to perform acquisition during AL for collecting gold labels.
3. The collected labels are used for training a resource-intensive teacher model, which has a higher quality than the distilled acquisition model.
4. The teacher model is used for pseudo-labeling of the whole unlabeled pool of instances.
5. Finally, we train a successor model of an arbitrary type on the dataset that contains automatically labeled instances and instances with gold labels obtained from human experts.

If the teacher model is expressive enough, it will generate reasonable pseudo labels, which can be reused by a model of any type and architecture. This additional annotation helps to mitigate the performance drop due to the acquisition-successor mismatch and to keep benefits of AL even when the successor model is more expressive than the model used for pseudolabeling. Meanwhile, PLASM helps to reduce the duration of AL iterations similarly to the approach of Shelmanov et al. (2021), and it does not introduce any additional computational overhead during the annotation process since training the teacher model and pseudo-labeling are performed after the AL annotation is completed.

3.4 Unlabeled Pool Subsampling

If the unlabeled pool of instances is large, which is a common situation, and a deep neural network is used as an acquisition model, the most time-consuming step of the AL cycle is the generation of predictions for unlabeled instances, which is necessary for uncertainty-based query strategies (refer to Table 1). We note that uncertainty estimates of the most certain instances in the unlabeled pool do not alter substantially across multiple AL iterations (Table 2). This means that AL wastes much time and resources on these unimportant instances. We claim that it is possible to recalculate uncertainty scores on the current iteration only for the top instances of the unlabeled pool, which were the most uncertain on previous iterations, while not sacrificing the benefits of AL.

We propose an unlabeled pool subsampling (UPS) algorithm, in which uncertainty estimates only for a fraction of instances are updated. Sampling of an instance on the current iteration is performed according to the Bernoulli distribution, which parameter depends on model uncertainty on previous iterations. Let u be the last recalculated uncertainty score of an instance on one of the previous iterations. We order the instances according to this value: $u_0 \leq u_1 \leq \cdots \leq u_i \leq \cdots \leq u_M$ and denote a normalized rank of an instance as $r_i = \frac{i}{M}$. Let $T > 0$ be a "temperature" hyperparameter and $\gamma \in [0, 1]$ be a hyperparameter that controls how many instances are always chosen. Then the probability of keeping an instance i for recalculation of uncertainty on the current iteration is:

$$\mathbb{P}(i) \propto \exp\left(-\frac{max(0, r_i - \gamma)}{T}\right).$$

Sampling certain instances with a non-negative probability instead of just ignoring them gives a chance of overcoming a situation when an informative instance is occasionally assigned a high certainty score and is never selected ever since.

On several initial iterations of AL, an acquisition model is trained on an extremely small amount of data, which leads to unreliable uncertainty estimates. To mitigate this problem, we suggest keeping the standard approach to performing instance acquisition on several first iterations and switch to the optimized process later during AL. We also note that interleaving the optimized selection with the standard approach, in which we recalculate the uncertainty for the whole unlabeled pool of instances, can help to keep the high performance of AL.

4 Experiments

4.1 Experimental Setup

We follow the common schema of AL experiments adopted in many previous works (Settles and Craven, 2008; Shen et al., 2017; Siddhant and Lipton, 2018; Shelmanov et al., 2021). We emulate the AL annotation cycle starting with a small random sample of the dataset used as a seed for the construction of the initial acquisition model. On each iteration, we pick a fraction of top instances from the unlabeled pool sorted using the query strategy and, instead of demonstrating them to annotators, automatically label them according to the gold standard. These instances are removed from the unlabeled pool and added to the training dataset for the next iterations. On each iteration, we train the successor model on the data acquired so far and evaluate it on the whole available test set. Acquisition and successor models are always trained from scratch. We run several iterations of emulation to build a chart, which demonstrates the performance of the successor depending on the amount of "labor" invested into the annotation process. To report standard deviations of scores, we repeat the whole experiment five times with different random seeds.

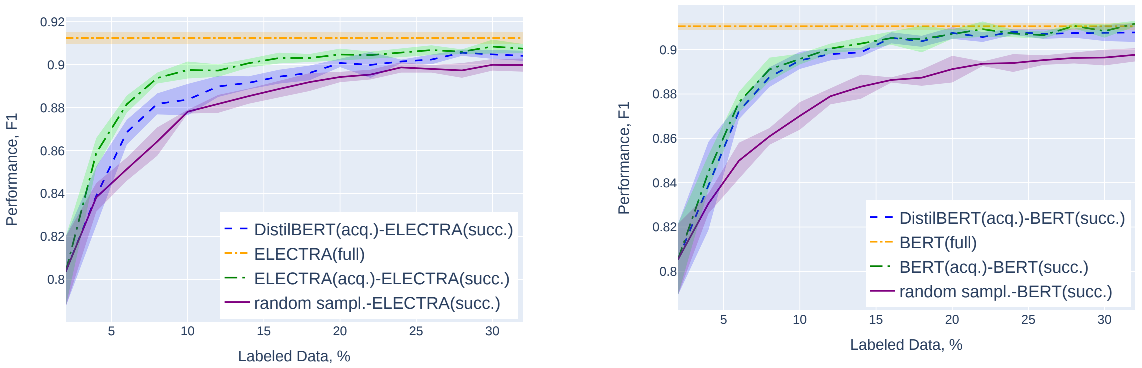
For classification, accuracy is used as the evaluation metric. For sequence tagging, we use the strict span-based F1-score (Sang and Meulder, 2003).

4.1.1 Datasets

We experiment with widely-used datasets for the evaluation of AL methods on text classification and sequence tagging tasks.

For text classification, we use the English AG News topic classification dataset (Zhang et al., 2015). We randomly select 1% of instances of the training set as a seed to train the initial acquisition model and select 1% of instances for "annotation" on each AL iteration.

For sequence tagging, we use English CoNLL-2003 (Sang and Meulder, 2003) and English



a) ELECTRA is a successor model.

b) BERT is a successor model.

Figure 2: AL experiments on CoNLL-2003, in which a successor model does not match an acquisition model (DistilBERT).

OntoNotes 5.0 (Pradhan et al., 2013). We randomly sample instances with a total number of tokens equal to 2% of all tokens from the training set as a seed. On each AL iteration, we select instances from the unlabeled pool until a total number of tokens equals 2% of all training tokens.

The corpora statistics are presented in Table 3 in Appendix A.

4.1.2 Model Choice, Training Details, and Hyperparameter Selection

We conduct experiments with pre-trained Transformers used in several previous works on AL: BERT (Devlin et al., 2019), ELECTRA (Clark et al., 2020), and DistilBERT (Sanh et al., 2019). In particular, we use the 'google/electra-base-discriminator' checkpoint from the Hugging Face repository (Wolf et al., 2020) for initialization of ELECTRA in both text classification and sequence tagging tasks. For initialization of DistilBERT and BERT for text classification, we take 'distilbert-base-uncased' and 'bert-base-uncased' checkpoints correspondingly. In experiments with sequence tagging, similar "cased" versions are used.

We keep a single pre-selected set of hyperparameters for all AL iterations. Tables 4, 5 in Appendix A describe the hyperparameter setup. Hyperparameter tuning on each AL iteration is very time-consuming. This is an important research problem but out of the scope of the current work.

4.2 Results and Discussion

4.2.1 Acquisition-Successor Mismatch

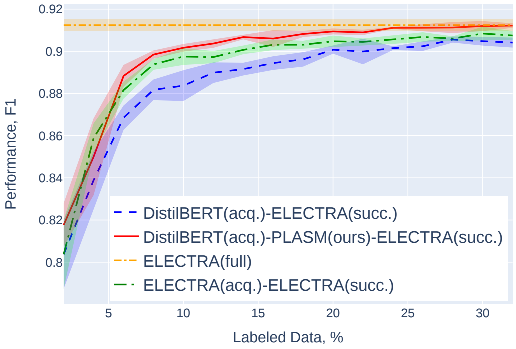


Figure 3: The performance of PLASM on CoNLL-2003 compared with the standard approach to AL.

First of all, we illustrate the acquisition-successor mismatch problem on the CoNLL-2003, OntoNotes, and AG News datasets (Figure 2a and Figures 5a, 6a in Appendix B). The presented results correspond to the findings of Lowell et al. (2019) and Shelmanov et al. (2021). We see a significant reduction in the performance of successor models when they are different from acquisition models (DistilBERT(acq.)-ELECTRA(succ.)) compared to the case when they are the same (ELECTRA(acq.)-ELECTRA(succ.)). The performance drop is especially notable on the CoNLL-2003 dataset in Figure 2a. The similar performance drop appears if we use BERT for acquisition and ELECTRA as a successor and vice versa (Figure 7 in Appendix B).

We show on both text classification and tagging tasks that replacing the original full-fledged acquisition model with its distilled version can alleviate this problem (Figure 2b, and Figures 5b, 6b in Appendix C). Previously, this effect was also revealed by Shelmanov et al. (2021) for tagging.

As we can see in Figure 2b, when DistilBERT is used as an acquisition model, the successor model based on BERT does not experience a performance drop. A similar effect can be noted for tagging on OntoNotes and for text classification on AG News.

Although we can mitigate the acquisition-successor mismatch problem for such pairs of models, it is still a serious constraint for applying AL. Obviously, such an approach is not feasible if there is no available distilled version of the model (e.g. there is no distilled version of ELECTRA). In the next section, we show that the proposed method based on pseudo-labeling helps to overcome this limitation and resolve the acquisition-successor mismatch problem in a more general case.

4.2.2 Pseudo-labeling for Acquisition-Successor Mismatch

Figure 3 and Figure 8 in Appendix C compare the performance of successor models constructed using the standard approach to AL, in which we use ELECTRA as an acquisition and successor model, and PLASM, in which we use DistilBERT for acquisition, BERT for pseudo-labeling, and ELECTRA as a successor. We can see that PLASM not only mitigates the acquisition-successor mismatch problem, but also helps to achieve slightly better results.

Figure 9a presents the results of the first ablation study, in which, for pseudolabeling, we replace BERT with a smaller model DistilBERT. The study demonstrates that in the case of acquisition-successor mismatch, using an expressive model (e.g. BERT) for pseudolabeling is necessary for achieving high scores and keeping AL useful in the beginning of annotation. Figure 9b presents the results of the second ablation study, in which, we use DistilBERT for acquisition and ELECTRA for pseudolabeling and as a successor. This study demonstrates that pseudolabeling on its own cannot alleviate the successor-mismatch completely. It is better to use an expressive pseudolabeling model that also matches the lightweight acquisition model (e.g. distilled model for acquisition, its teacher – for labeling), as it is proposed in PLASM. Figure 10 in Appendix C shows that PLASM also effectively mitigates performance drop due to mismatch between a DistilBERT acquisition model and a CNN-BiLSTM-CRF successor model.

Table 1 and Table 6 in Appendix D summarize the time required for conducting AL iterations with different acquisition functions on the AG News and CoNLL-2003 datasets. As we can see, since PLASM uses DistilBERT for acquisition, our method reduces the iteration time by more than 30% compared to the standard approach, in which ELECTRA is used for acquisition. Thereby, empirical results show that PLASM offers two benefits: (1) it helps to improve the performance of the final successor model that uses data obtained with AL; (2) it reduces the time of an AL iteration and required computational resources for training and running acquisition models. These benefits substantially increase the practicality of using AL in interactive annotation tools.

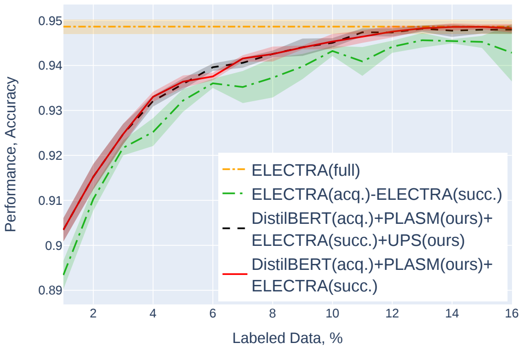


Figure 4: The performance of UPS on AG News compared with baselines ($\gamma = 0.1$, $T = 0.01$).

4.2.3 Unlabeled Pool Subsampling

Table 1 compares the duration of AL iterations on the AG News dataset, including the duration of the acquisition model training step and the duration of inference on instances from the unlabeled pool. We can see that the inference step is very time-consuming, especially on early iterations, and takes more than half of the time required for performing an AL iteration. Therefore, we claim that in such cases, it is more important to accelerate the inference step rather than the training step as it was done in previous work (Shen et al., 2017).

To justify our approach to accelerating the inference step, we show that many unlabeled instances have similar uncertainty estimates across different AL iterations. Table 2 presents the fraction of instances, which would be standardly queried on the current iteration if we selected them from the whole unlabeled pool that are contained in k-% of most uncertain instances, according to the acquisition model built on the previous AL iteration. For example, we observe that 50% of the most uncertain instances according to the model trained on the

		ELECTRA	**BERT**	**DistilBERT**	**ELECTRA with UPS (ours)**	**DistilBERT with UPS (ours)**
Iter. 2	Train	176.3 ± 1.4	174.8 ± 1.4	87.4 ± 0.8	178.0 ± 1.4	87.9 ± 0.5
	Inference	622.2 ± 9.4	623.8 ± 7.5	481.8 ± 17.2	630.9 ± 12.3	483.2 ± 23.0
	Overall	798.6 ± 9.6	798.6 ± 8.4	569.2 ± 17.5	808.8 ± 12.8	571.1 ± 22.6
Iter. 6	Train	342.8 ± 5.7	339.9 ± 4.2	174.1 ± 2.9	342.2 ± 5.3	173.0 ± 1.4
	Inference	600.5 ± 10.4	596.4 ± 6.6	455.1 ± 8.9	**58.9±3.3**	**50.0±6.4**
	Overall	943.4 ± 15.9	936.3 ± 8.8	629.1 ± 9.7	**401.1±3.4**	**222.9±5.9**
Iter. 10	Train	504.6 ± 6.3	498.8 ± 3.9	257.5 ± 3.9	502.7 ± 6.0	255.1 ± 3.4
	Inference	573.0 ± 6.9	577.5 ± 7.7	434.6 ± 4.6	**55.5±2.9**	**42.6±7.1**
	Overall	1077.6 ± 13.1	1076.4 ± 10.9	692.1 ± 5.5	**558.2±4.4**	**297.7±10.3**
Iter. 15	Train	701.9 ± 7.2	714.9 ± 20.5	358.3 ± 3.0	704.8 ± 11.7	359.3 ± 5.4
	Inference	548.6 ± 9.2	541.0 ± 5.0	415.9 ± 10.2	**59.4±3.1**	**39.3±2.6**
	Overall	1250.5 ± 16.0	1255.9 ± 18.4	774.2 ± 10.8	**764.2±10.6**	**398.6±6.8**
Overall train		6323.7 ± 72.1	6294.8 ± 73.7	3215.1 ± 38.5	6333.3 ± 92.8	3204.5 ± 32.5
Overall inference		8799.2 ± 150.7	8787.5 ± 102.7	6682.1 ± 96.2	**3110.9±85.3**	**2332.2±86.2**
Overall		15122.9 ± 213.4	15082.2 ± 141.1	9897.1 ± 112.8	**9444.2±113.6**	**5536.7±100.8**

Table 1: Duration of training and inference steps of AL iterations in seconds on AG News. Hardware configuration: 2 Intel Xeon Platinum 8168, 2.7 GHz, 24 cores CPU; NVIDIA Tesla v100 GPU, 32 Gb of VRAM.

Top-k% / Curr. AL iter.	**1**	**2**	**6**
10%	0.503	0.649	0.924
20%	0.789	0.883	0.992
30%	0.915	0.947	0.995
40%	0.958	0.976	1.000
50%	0.980	0.991	1.000

Table 2: A fraction of instances that would be standardly selected on the current AL iteration, contained in top-k% uncertain instances according to the acquisition model on the previous iteration (AG News corpus).

first iteration contains more than 99% of instances from the "standard query" on the second iteration, and 30% contain almost 95% of instances from the "standard query". Later iterations have even a better trade-off. Thereby, it is reasonable to avoid spending computational resources on instances that were most certain in previous iterations.

If we exclude a big part of the unlabeled pool from consideration during acquisition, the benefits of AL can potentially deteriorate. Results of experiments presented in Figure 4 and Figures 11, 12 in Appendix D show that the proposed UPS algorithm does not lead to the performance drop compared to the standard approach, in which we consider the whole unlabeled pool for instance selection. Meanwhile, the results of the ablation study in Figure 13 (Appendix D) demonstrate that the baseline, which randomly subsamples the unlabeled dataset, has a performance drop compared to UPS.

From Table 1, we can see that UPS accelerates the query process up to 10 times. The corresponding results for CoNLL-2003 are presented in Table 6 in Appendix D. Overall, applying both PLASM and UPS algorithms on AG News reduces the duration of AL iterations by more than 60% comparing with the standard approach. We can also tune the hyperparameters γ and T to reduce duration further in exchange for slightly worse scores.

5 Conclusion

We investigated several obstacles to deploying AL in practice and proposed two algorithms that help to overcome them. In particular, we considered the acquisition-successor mismatch problem revealed by Lowell et al. (2019), as well as the problem related to the excessive duration of AL iterations with uncertainty-based query strategies and deep learning models. We demonstrate that the proposed PLASM algorithm helps to deal with both of these issues: it removes the constraint on the type of the successor model trained on the data labeled with AL and allows the use of lightweight acquisition models that have good training and inference performance, as well as a small memory footprint. The unlabeled pool subsampling algorithm helps to substantially decrease the inference time during AL without a loss in the quality of successor models. Together the PLASM and UPS algorithms help reduce the duration of an AL iteration by more than 60%. We consider that the conducted empirical investigations and the proposed methods will help to increase the practicality of using deep AL in interactive annotation tools.

There are still many issues that hinder the application of AL techniques. We consider that one of the most important obstacles is the necessity of hyperparameter optimization of deep learning models that can take a prohibitively long time to keep the annotation process interactive. We are looking forward to addressing this problem in future work.

References

Jason Baldridge and Miles Osborne. 2004. Active learning and the total cost of annotation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 9–16, Barcelona, Spain. Association for Computational Linguistics.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural network. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1613–1622. JMLR.org.

Kianté Brantley, Hal Daumé III, and Amr Sharaf. 2020. Active imitation learning with noisy guidance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2093–2105, Online. Association for Computational Linguistics.

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*.

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

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.

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.

Alexander Erdmann, David Joseph Wrisley, Benjamin Allen, Christopher Brown, Sophie Cohen-Bodénès, Micha Elsner, Yukun Feng, Brian Joseph, Béatrice Joyeux-Prunel, and Marie-Catherine de Marneffe. 2019. Practical, efficient, and customizable active learning for named entity recognition in the digital humanities. 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 2223–2234, Minneapolis, Minnesota. Association for Computational Linguistics.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, Copenhagen, Denmark. 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 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1050–1059.

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

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

Rong Hu, Brian Mac Namee, and Sarah Jane Delany. 2016. Active learning for text classification with reusability. *Expert Systems with Applications: An International Journal*, 45(C):438–449.

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. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 3–12. ACM/Springer.

Ming Liu, Wray Buntine, and Gholamreza Haffari. 2018. Learning how to actively learn: A deep imitation learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1874–1883, Melbourne, Australia. Association for Computational Linguistics.

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.

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.

Katerina Margatina, Loic Barrault, and Nikolaos Aletras. 2021. Bayesian active learning with pretrained language models. *arXiv preprint arXiv:2104.08320*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Ameya Prabhu, Charles Dognin, and Maneesh Singh. 2019. Sampling bias in deep active classification: An empirical study. 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 4049–4059.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. 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. *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Burr Settles. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

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.

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 Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 252–256, Vancouver, Canada. Association for Computational Linguistics.

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.

Katrin Tomanek and Katherina Morik. 2011. Inspecting sample reusability for active learning. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 169–181. JMLR Workshop and Conference Proceedings.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Thuy-Trang Vu, Ming Liu, Dinh Phung, and Gholamreza Haffari. 2019. Learning how to active learn by dreaming. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4091–4101, Florence, Italy. 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.

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.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, Neural Information Processing Systems'15, page 649–657, Cambridge, MA, USA. MIT Press.

A Dataset Statistics and Model Hyperparameters

Table 3: Dataset statistics. We provide a number of sentences/tokens for the training and test sets. k stands for a size of seeding datasets (% of the training dataset) and a size of sets of instances selected for “annotation” on each iteration. C is a number of classes/entity types.

Datasets	**Train**	**Test**	**k**	**C**
CoNLL-2003	15K/203.6K	3.7K/46.4K	2%	4(5)
OntoNotes 5.0	59.9K/1088.5K	8.3K/152.7K	2%	18
AG News	120K/4556.4K	7.6K/287.6K	1%	4

Table 4: 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.

Hparam	**AG News**	**CoNLL**	**OntoNotes**
Number of epochs	5	5	5
Batch size	16	16	16
Minimum number of steps	350	350	350
Max sequence length	256	-	-
Optimizer	Adam	Adam	Adam
Learning rate	2e-5	5e-5	5e-5
Weight decay	0.01	0.01	0.01
Gradients clipping	1.	1.	1.
Scheduler	STLR	STLR	STLR
% of warmup steps	0.1	0.1	0.1

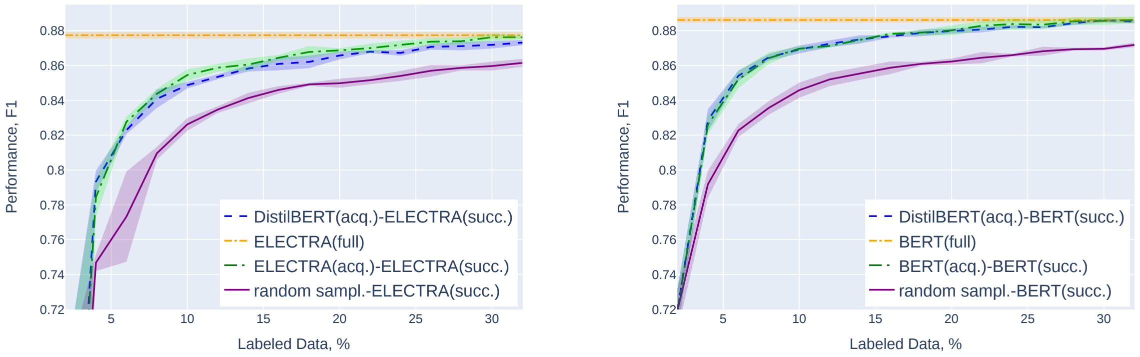
Table 5: Hyperparameter values of the CNN-BiLSTM-CRF model.

Hparam	**ConLL-2003**
Word embeddings pre-trained model	GloVe (Pennington et al., 2014) [1]
Word embedding dim.	100
Char embedding dim.	25
CNN dim.	30
CNN filters	[2, 3]
CNN activation	Mish
RNN num. layers	1
RNN hidden size	128
RNN recur. dropout prob.	0.1
RNN layer dropout prob.	0.1
Encoder dropout prob	0.1
Feed forward num. layers	1
Feed forward hidden size	128
Feed forward activation	Tanh
Feed forward dropout prob.	0.1

---

[1] https://flair.informatik.hu-berlin.de/resources/embeddings/token/glove.gensim

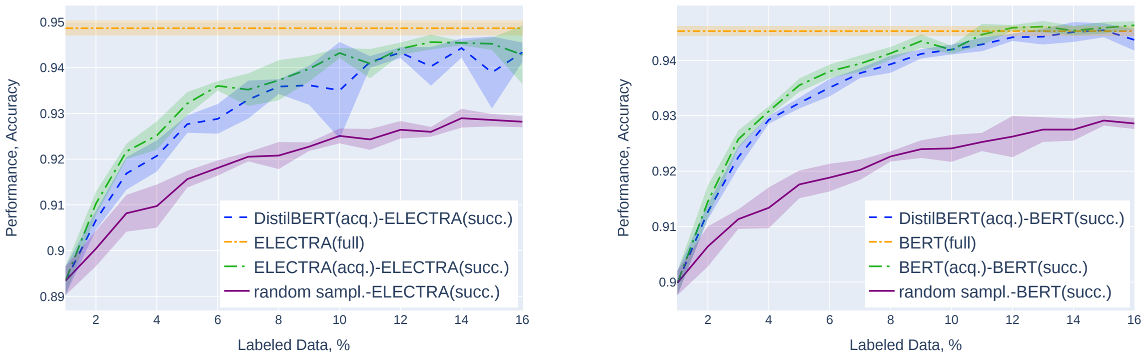
B Additional Experimental Results with Acquisition-successor Mismatch



a) ELECTRA is a successor model.

b) BERT is a successor model.

Figure 5: AL experiments on OntoNotes, in which a successor model does not match an acquisition model (DistilBERT).



a) ELECTRA is a successor model.

b) BERT is a successor model.

Figure 6: AL experiments on AG News, in which a successor model does not match an acquisition model (DistilBERT).

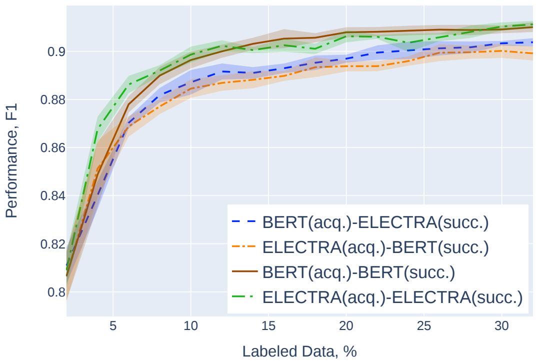
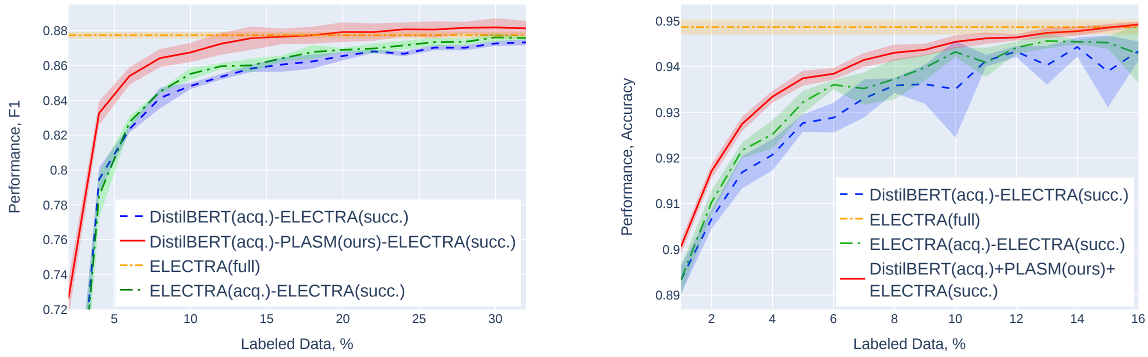


Figure 7: AL experiments on CoNLL-2003, in which a successor model does not match an acquisition model. This experiment demonstrates that models with similar expressiveness and size (BERT and ELECTRA) cannot be used interchangeably in AL.

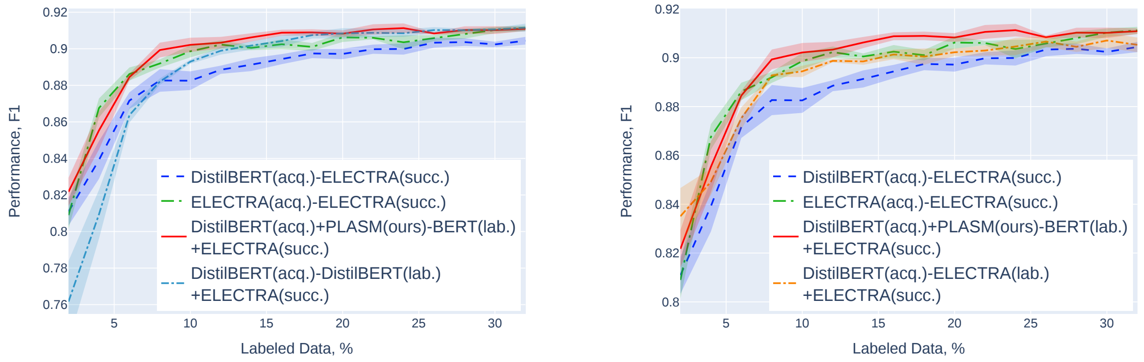
C Additional Experimental Results with PLASM



a) OntoNotes dataset.

b) AG News dataset.

Figure 8: The performance of PLASM compared with the standard approach to AL on OntoNotes and AG News.



a) DistilBERT for pseudolabeling.

b) ELECTRA for pseudolabeling.

Figure 9: Ablation studies of PLASM on the CoNLL-2003 dataset, in which inappropriate model is used for pseudolabeling.

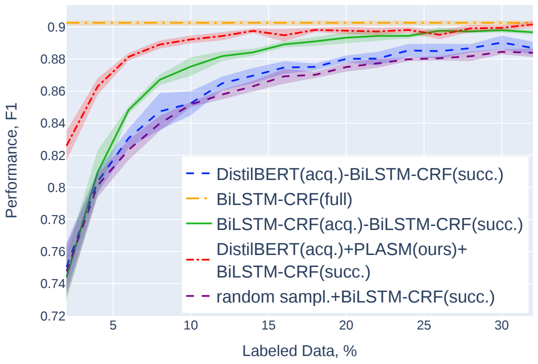
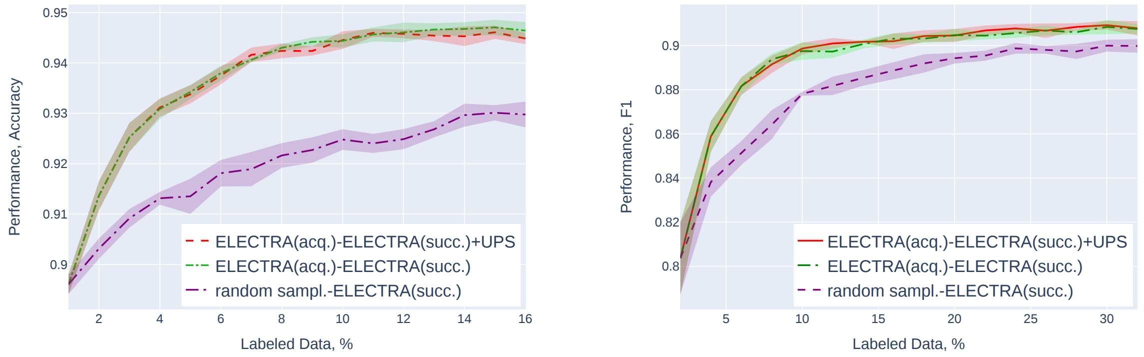


Figure 10: Experiments with PLASM and standard approaches, in which BiLSTM-CRF is used as a successor model. We can see that due to using PLASM and the expressiveness of the labeling model (BERT), the successor achieves substantial improvements over the baseline.

D Additional Experimental Results with UPS



a) AG News dataset.

b) CoNLL-2003 dataset.

Figure 11: The performance of UPS compared with the standard approach to AL on AG News and CoNLL-2003 datasets with ELECTRA as a successor model ($\gamma = 0.1$, $T = 0.01$).

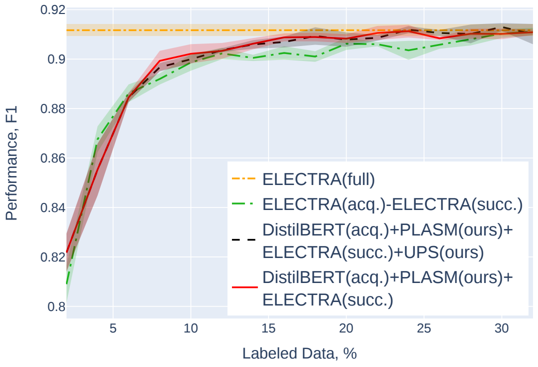


Figure 12: The performance of UPS in conjunction with PLASM on CoNLL-2003 compared with baselines ($\gamma = 0.1$, $T = 0.01$).

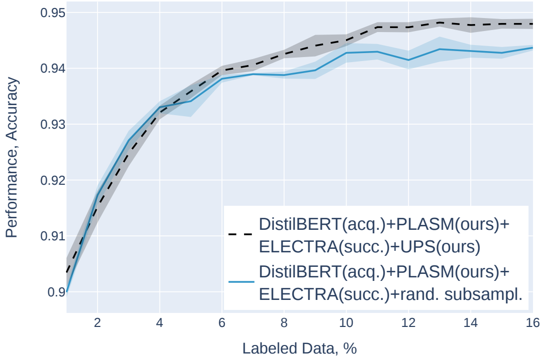


Figure 13: The comparison of UPS with a random-subsampling baseline on the AG News dataset ($\gamma = 0.1$, $T = 0.01$).

		ELECTRA	BERT	DistilBERT	**ELECTRA with UPS (ours)**	**DistilBERT with UPS (ours)**
Iter. 2	Train	44.8 ± 0.3	50.9 ± 1.6	29.1 ± 0.3	43.3 ± 0.8	26.4 ± 2.5
	Inference	25.9 ± 0.3	25.9 ± 0.3	19.6 ± 0.3	25.7 ± 0.4	19.9 ± 0.9
	Overall	70.6 ± 0.6	76.8 ± 1.7	48.7 ± 0.5	69.0 ± 1.0	46.3 ± 3.1
Iter. 6	Train	74.9 ± 1.6	81.4 ± 1.4	49.7 ± 1.3	66.9 ± 1.6	44.2 ± 4.0
	Inference	23.8 ± 0.0	23.4 ± 0.3	17.9 ± 0.0	**3.2±0.2**	**2.3±0.2**
	Overall	98.6 ± 1.5	104.8 ± 1.1	67.5 ± 1.4	**70.1±1.6**	**46.5±4.2**
Iter. 10	Train	95.6 ± 1.1	105.7 ± 1.5	63.6 ± 2.0	88.4 ± 1.2	57.1 ± 5.5
	Inference	21.3 ± 0.1	21.4 ± 0.2	15.9 ± 0.5	**2.6±0.2**	**2.3±0.1**
	Overall	116.9 ± 1.2	127.1 ± 1.5	79.5 ± 2.4	**91.0±1.3**	**59.4±5.6**
Iter. 15	Train	122.2 ± 1.2	133.4 ± 3.1	79.0 ± 1.3	129.9 ± 3.2	74.6 ± 6.4
	Inference	18.9 ± 0.2	18.6 ± 0.1	14.0 ± 0.2	**2.0±0.1**	**1.4±0.1**
	Overall	141.1 ± 1.0	151.9 ± 3.2	92.9 ± 1.2	**131.9±3.1**	**76.0±6.5**
Overall train		1266.6 ± 16.9	1387.1 ± 26.3	838.6 ± 19.2	1195.0 ± 25.0	748.3 ± 70.4
Overall inference		339.1 ± 3.5	335.5 ± 4.7	252.9 ± 3.9	**128.9±5.6**	**97.5±5.1**
Overall		1605.7 ± 18.8	1722.6 ± 24.1	1091.4 ± 18.4	**1323.9±28.5**	**845.8±75.1**

Table 6: Duration of training and inference steps of AL iterations in seconds on CoNLL-2003. We highlight with the bold font the values affected by UPS. Hardware configuration: 2 Intel Xeon Platinum 8168, 2.7 GHz, 24 cores CPU; NVIDIA Tesla v100 GPU with 32 Gb of VRAM.