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 annotating, we have to build an acquisition model almost without insights from the data that could help us to do feature engineering or to introduce 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 followers 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 one 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 (ASM) raises the question of whether AL is a practical technique at all since the usage of different models on the annotated dataset is a common situation. The problem is complicated by a contradiction between the fact that the acquisition model is required to be as lightweight as possible to mitigate computational overhead and the successor model should 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 ASM 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 Pseudo-Labeling for Acquisition Successor Mismatch (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 ASM problem. It helps to reduce the hardware requirements and the duration of AL iterations.

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).

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.

A few works have experimented with Bayesian query strategies for AL. Shen et al. (2017), Sid-
dhant 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 the 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 big 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. However, this requires an excessive amount of computation while the transferability of learned policies across domains and tasks is underresearched.

Finally, Lowell et al. (2019) question the usefulness of AL techniques in general. They demonstrate that due to the ASM problem, AL can be even detrimental to the performance of the successor. This finding is also revealed for classical machine learning models by Baldrige 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 ASM 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:

$$LC_{cls} = 1 - \max_y P(y|x).$$

**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:

$$MNLP_{net} = 1 - \frac{1}{n} \max_{y_1,\ldots,y_n} \frac{1}{n} \sum_{i=1}^{n} \log P[y_i | \{y_j \setminus y_i, \{x_j\}]]$$

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).

**Mahalanobis Distance (MD)** between a test instance and the closest class-conditional Gaussian distribution is suggested by Lee et al. (2018) for detection of out-of-distribution instances and adversarial attacks. MD is a strong baseline for uncertainty estimation of NLP model predictions (Podolsky et al., 2021) and is also a backbone for other subsequent techniques (Zhou et al., 2021). We use it as an informativeness score in AL since previous work shows that MD captures epistemic uncertainty well (Podolsky et al., 2021).

$$MD_{cls} = \min_{c \in C} (h_i - \mu_c)^T \Sigma^{-1} (h_i - \mu_c), \quad (1)$$

where $h_i$ is a hidden representation of a $i$-th instance, $\mu_c$ is a centroid of a class $c$, and $\Sigma$ is a covariance matrix for hidden representations of training instances.
3.2 Models

We use the standard models based on the Transformer architecture (Vaswani et al., 2017): BERT, ELECTRA, and XLNet (Yang et al., 2019). 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): \( L_{distil} = - \sum_{i,c} t_{ic} \log (s_{ic}) \), 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 with a 40% reduction of a memory footprint. It 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 consists of the following steps:

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. The automatically acquired annotations are filtered to reduce noise introduced by mistakes of the pseudolabeling model. In the main experiments, we use TracIn – a strong and practical method for mislabelled data identification (Pruthi et al., 2020). In the ablation study, we also test a simpler solution: filtering instances with high uncertainty of the pseudolabeling model predictions. The fraction of the filtered out instances in both cases is determined from the evaluation score of the pseudolabeling model on a held-out subset of the training corpus (100%-score).

6. 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 filtered and reuses by another model of a different type and architecture. This additional annotation helps to mitigate the performance drop due to ASM 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 itera-
We order the instances according to this value: 

\[ \text{We claim that it is possible to recalculate uncertainty scores on the current iteration only for the top instances of the unlabeled pool, which were the most uncertain on previous iterations, while not sacrificing the benefits of AL.} \]

We propose an unlabeled pool subsampling (UPS) algorithm, in which uncertainty estimates only for a fraction of instances are updated. On the current iteration, we suggest always selecting a fraction of the most uncertain instances on previous iterations equal to \( \gamma \in [0, 1] \) and sample a small portion of instances with a probability that depends on their rank in a list sorted by their uncertainty. Formally, this can be written as follows. 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. Then the probability of keeping an instance \( i \) for recalculation of uncertainty on the current iteration is:

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

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 uncertainty score and is never selected ever since. This method is inspired by subsampling techniques used in gradient boosting algorithms for selecting a training subset for decision trees (Ke et al., 2017; Ibragimov and Gusev, 2019).

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 switching 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. In most experiments, we use LC or MNLP query strategies for classification and sequence tagging correspondingly. Results with MD are presented only in Figure 7 in Appendix B.

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 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.
4.1.2 Model Choice, Training Details, and Hyperparameter Selection

We conduct experiments with pre-trained Transformers used in several previous works on AL. The exact checkpoints and parameter numbers are presented in Table 5 in Appendix A.

We keep a single pre-selected set of hyperparameters for all AL iterations. Tables 4, 6 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

First of all, we illustrate the ASM problem on the CoNLL-2003, OntoNotes, and AG News datasets (Figure 1a and Figures 4a, 5a 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 1a. The similar performance drop appears if we use BERT for acquisition and ELECTRA as a successor and vice versa (Figure 6 in Appendix B). The ASM problem appears to be even more severe with the modern uncertainty estimation technique based on MD. Figure 7 in Appendix B shows the results of experiments with MD on the AG News dataset. Training the ELECTRA successor on data acquired during AL with MD and DistilBERT results in lower performance than when using data acquired with LC.

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 1b, and Figures 4b, 5b in Appendix C). Previously, this effect was also revealed by Shelmanov et al. (2021) for tagging. As we can see in Figure 1b, 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 ASM 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
The proposed method based on pseudo-labeling helps to overcome this limitation and resolve the ASM problem in a more general case.

4.2.2 Pseudo-labeling for Acquisition-Successor Mismatch

Figure 2 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 confidently mitigates the ASM problem. Figure 10 in Appendix C shows that PLASM also effectively mitigates performance drop due to ASM between a DistilBERT acquisition model and a CNN-BiLSTM-CRF successor model. Figure 11 in Appendix C illustrates the same finding for the case when XLNet is used as a successor.

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 ASM, 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 pseudo-labeling on its own cannot alleviate the ASM 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.

The ablation study of the methods for filtering erroneous instances in the pseudo-labeling step is conducted on the AG News dataset in Figures 12a,b in Appendix C. Applying each of the methods gives substantial improvements over the PLASM without the filtering step, while TracIn is slightly better than thresholding uncertainty of pseudo-labeling model predictions. We note that for XLNet as a successor, PLASM without filtering alleviates the ASM problem, but does not approach the performance of the case when XLNet is used as an acquisition model.

Table 1 and Table 7 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 alleviate the ASM problem in 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.

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 first iteration contains more than 99% of instances from the “standard query” on the second iteration, and 30% contains almost 95% of instances from the “standard query”. Later iterations have even a better trade-off. Thereby, it is reasonable to avoid
while, the results of the ablation study in Figure 15 demonstrate that the baseline, which randomly subsamples the unlabeled dataset, has a performance drop compared to UPS. In another ablation study, we set $T = 0$, which means that UPS just takes a fraction of the most uncertain instances (Figure 16). On some iterations, this results in a slight reduction of performance.

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 7 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 $\gamma$ 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


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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Test</th>
<th>(k)</th>
<th>(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoNLL-2003</td>
<td>15K/203.6K</td>
<td>3.7K/46.4K</td>
<td>2%</td>
<td>4(5)</td>
</tr>
<tr>
<td>OntoNotes 5.0</td>
<td>59.9K/1088.5K</td>
<td>8.3K/152.7K</td>
<td>2%</td>
<td>18</td>
</tr>
<tr>
<td>AG News</td>
<td>120K/4556.4K</td>
<td>7.6K/287.6K</td>
<td>1%</td>
<td>4</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Hparam</th>
<th>AG News</th>
<th>CoNLL</th>
<th>OntoNotes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of epochs</td>
<td>5</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Batch size</td>
<td>16</td>
<td>16</td>
<td>1000</td>
</tr>
<tr>
<td>Minimum number of steps</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Max sequence length</td>
<td>256</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Optimizer</td>
<td>AdamW</td>
<td>AdamW</td>
<td>AdamW</td>
</tr>
<tr>
<td>Learning rate</td>
<td>2e-5</td>
<td>5e-5</td>
<td>5e-5</td>
</tr>
<tr>
<td>Weight decay</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>Gradients clipping</td>
<td>1.</td>
<td>1.</td>
<td>1.</td>
</tr>
<tr>
<td>% of warmup steps</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 5: Transformers model checkpoints from HuggingFace repository.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Checkpoint</th>
<th># Param.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG-News</td>
<td>XLNet</td>
<td>xlnet-base-cased</td>
<td>117M</td>
</tr>
<tr>
<td></td>
<td>BERT</td>
<td>bert-base-uncased</td>
<td>110M</td>
</tr>
<tr>
<td></td>
<td>DistilBERT</td>
<td>distilbert-base-uncased</td>
<td>66M</td>
</tr>
<tr>
<td></td>
<td>ELECTRA</td>
<td>google/electra-base-discriminator</td>
<td>110M</td>
</tr>
<tr>
<td>CoNLL-2003 / OntoNotes 5.0</td>
<td>ELECTRA</td>
<td>google/electra-base-discriminator</td>
<td>110M</td>
</tr>
</tbody>
</table>

Table 6: Hyperparameter values of the CNN-BiLSTM-CRF model.

<table>
<thead>
<tr>
<th>Hparam</th>
<th>ConLL-2003</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word embeddings</td>
<td>GloVe (Pennington et al., 2014)</td>
</tr>
<tr>
<td>pre-trained model</td>
<td></td>
</tr>
<tr>
<td>Word embedding dim.</td>
<td>100</td>
</tr>
<tr>
<td>Char embedding dim.</td>
<td>30</td>
</tr>
<tr>
<td>CNN dim.</td>
<td>30</td>
</tr>
<tr>
<td>CNN filters [2, 3]</td>
<td></td>
</tr>
<tr>
<td>RNN num. layers</td>
<td>1</td>
</tr>
<tr>
<td>RNN hidden size</td>
<td>200</td>
</tr>
<tr>
<td>RNN word dropout prob.</td>
<td>0.3</td>
</tr>
<tr>
<td>RNN locked dropout prob.</td>
<td>0.2</td>
</tr>
<tr>
<td>Encoder dropout prob.</td>
<td>0.0</td>
</tr>
<tr>
<td>Feed forward num. layers</td>
<td>1</td>
</tr>
<tr>
<td>Feed forward hidden size</td>
<td>200</td>
</tr>
<tr>
<td>Feed forward activation</td>
<td>Tanh</td>
</tr>
<tr>
<td>Feed forward dropout prob.</td>
<td>0.0</td>
</tr>
<tr>
<td>Batch size</td>
<td>32</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.015</td>
</tr>
<tr>
<td>Momentum</td>
<td>0.9</td>
</tr>
<tr>
<td>Number of epochs</td>
<td>50</td>
</tr>
<tr>
<td>Optimizer</td>
<td>SGD</td>
</tr>
<tr>
<td>Gradients clipping</td>
<td>5</td>
</tr>
</tbody>
</table>
**B Additional Experimental Results with Acquisition-successor Mismatch**

Figure 4: AL experiments on OntoNotes, in which a successor model does not match an acquisition model (DistilBERT).

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

Figure 6: 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.
Figure 7: AL experiments on AG News with Mahalanobis distance used as an uncertainty measure. This experiment demonstrates that the acquisition-successor mismatch problem also persists for this modern uncertainty estimation technique.

C Additional Experimental Results with PLASM

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

Figure 9: Ablation studies of PLASM on the CoNLL-2003 dataset, in which an inappropriate model is used for pseudolabeling.
Figure 10: Experiments with PLASM and standard approaches, in which CNN-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.

Figure 11: Experiments with PLASM and standard approaches on AG News, in which XLNet is used as a successor model.

Figure 12: Ablation studies of filtering methods in PLASM on the AG News dataset.

a) ELECTRA as a successor.

b) XLNet as a successor.
D Additional Experimental Results with UPS

Figure 13: 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 14: The performance of UPS in conjunction with PLASM on CoNLL-2003 compared with baselines ($\gamma = 0.1, T = 0.01$).

Figure 15: The comparison of UPS with a random-subsampling baseline on the AG News dataset ($\gamma = 0.1, T = 0.01$).
Table 7: 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.

![Graph showing performance and accuracy variations](image)

Figure 16: Ablation study for the parameter $T$ in the UPS algorithm. We see that when sampling a total of 5% of the dataset to select for the query, using a non-zero value for the parameter $T$ gives an increase in performance compared to a case when only 5% most uncertain samples are considered for query (i.e. $T = 0$).