Bayesian Active Learning with Pretrained Language Models

Katerina Margatina Loic Barrault Nikolaos Aletras
Computer Science Department, University of Sheffield
{k.margatina,l.barrault,n.aletras}@sheffield.ac.uk

Abstract

Active Learning (AL) is a method to iteratively select data for annotation from a pool of unlabeled data, aiming to achieve better model performance than random selection. Previous AL approaches in Natural Language Processing (NLP) have been limited to either task-specific models that are trained from scratch at each iteration using only the labeled data at hand or using off-the-shelf pretrained language models (LMs) that are not adapted effectively to the downstream task. In this paper, we address these limitations by introducing BALM; Bayesian Active Learning with pretrained language Models. We first propose to adapt the pretrained LM to the downstream task by continuing training with all the available *unlabeled* data and then use it for AL. We also suggest a simple yet effective fine-tuning method to ensure that the adapted LM is properly trained in both low and high resource scenarios during AL. We finally apply Monte Carlo dropout to the downstream model to obtain well-calibrated confidence scores for data selection with uncertainty sampling. Our experiments in five standard natural language understanding tasks demonstrate that BALM provides substantial data efficiency improvements compared to various combinations of acquisition functions, models and fine-tuning methods proposed in recent AL literature.

1 Introduction

Active Learning (AL) is a method for training supervised models in a data-efficient way (Cohn et al., 1996; Settles, 2009). AL methods iteratively alternate between (i) model training with the labeled data available; and (ii) data selection for annotation using a stopping criterion, e.g. until exhausting a fixed annotation budget or reaching a pre-defined performance on a held-out dataset. Data selection is performed by an acquisition function that ranks unlabeled data points by some *informativeness* metric aiming to improve over random selection.

AL has been used in NLP for part-of-speech tagging (Engelson and Dagan, 1996), parsing (Tang et al., 2002), sentiment analysis (Li et al., 2012), machine translation (Haffari et al., 2009) and quality estimation (Beck et al., 2013) among others. It is especially useful in scenarios where a large pool of unlabeled data is available but only a limited annotation budget can be afforded; or where expert annotation is prohibitively expensive and time consuming.

Traditional Bayesian AL methods use uncertainty sampling (i.e. informativeness is measured by predictive uncertainty) and typically require probabilistic machine learning models to acquire good uncertainty estimates for the candidate data points. However, current work uses deep learning models that provide large performance gains but not well-calibrated confidence scores (Guo et al., 2017), i.e. predictive softmax probabilities are erroneously interpreted as model confidence (Gal and Ghahramani, 2016). Several approaches have been proposed to calibrate the output probability distribution of deep neural networks, such as temperature scaling (Guo et al., 2017), Monte Carlo dropout (Gal and Ghahramani, 2016) and model ensembles (Lakshminarayanan et al., 2017). Using uncertainty sampling with the vanilla output probabilities for AL may lead to incorrect conclusions, i.e. poor results may be attributed to the acquisition method, while the problem may be in fact the lack of calibration. Still, only a few deep Bayesian AL approaches apply a calibration method to the posterior probabilities (Gal et al., 2017; Shen et al., 2017; Siddhant and Lipton, 2018; Lowell and Lipton, 2019; Ein-Dor et al., 2020).

Furthermore, most current AL approaches in NLP use task-specific neural models that are trained from scratch at each iteration (Shen et al., 2017; Siddhant and Lipton, 2018; Prabhu et al., 2019; Ikhwantri et al., 2018; Kasai et al., 2019). However, task-specific models are usually out-

performed by pretrained language models (LMs) adapted to end-tasks (Howard and Ruder, 2018; Devlin et al., 2019), making them suboptimal for AL. Only recently, pretrained LMs such as BERT (Devlin et al., 2019) have been introduced in AL settings (Yuan et al., 2020; Ein-Dor et al., 2020), where they are transferred and used as downstream classification models. Still, they are trained at each AL iteration with a standard fine-tuning approach that mainly includes a pre-defined number of training epochs, which has been demonstrated to be unstable, especially in small datasets (Mosbach et al., 2021; Zhang et al., 2020; Dodge et al., 2020). Since AL includes both low and high data resource settings, the AL model training scheme should be robust in both scenarios.[1]

To address these limitations, we introduce Bayesian Active Learning with pretrained language Models (BALM). Contrary to previous work (Yuan et al., 2020; Ein-Dor et al., 2020) that also use BERT (Devlin et al., 2019), our proposed method accounts for the varying data availability settings, the instability of fine-tuning and the poor calibrated confidence scores for data selection:

1. We propose to continue *pretraining* the LM with the available *unlabeled* data to adapt it to the task-specific domain. This way, we leverage not only the available labeled data at each AL iteration, but the entire unlabeled pool;

2. We further propose a simple yet effective fine-tuning method that is robust in both low and high resource data AL settings;

3. We improve data acquisition by providing well-calibrated uncertainty estimates by using Monte Carlo dropout (Gal and Ghahramani, 2016) instead of using the softmax output as confidence scores.

We evaluate BALM on five standard natural language understandings tasks using a full suite of uncertainty-based acquisition functions, and compare against strong baselines that are based on diversity sampling (i.e. BERT K-means clustering), both uncertainty and diversity (e.g. BADGE (Ash et al., 2020), ALPS (Yuan et al., 2020),), and random sampling. We show that BALM outperforms all combinations of acquisition functions and training methods across all datasets (§5). We also find that our proposed training strategy yields substantial performance improvement when combined with any acquisition function (§6).

[1]During the first few AL iterations the available labeled data is limited (*low-resource*), while it could become very large towards the last iterations (*high-resource*).

2 Background and Related Work

2.1 Problem Formulation

Given a downstream classification task with C classes, a typical pool-based AL setup consists of a pool of unlabeled data $\mathcal{D}_{\textbf{pool}}$, a model $\mathcal{M}$, a pre-defined annotation budget b of data points and an acquisition function $a(.)$ for selecting k unlabeled data points for annotation (i.e. acquisition size) until b runs out. A validation set $\mathcal{D}_{\textbf{val}}$ is used to evaluate $\mathcal{M}$ after each iteration. The goal is to achieve data efficiency by selecting the least number of data points from $\mathcal{D}_{\textbf{pool}}$ for annotation and achieve the highest performance on the validation set $\mathcal{D}_{\textbf{val}}$ (Siddhant and Lipton, 2018). The performance of the algorithm is assessed by training a model on the actively acquired dataset and evaluating on a held-out test set $\mathcal{D}_{\textbf{test}}$.

AL systems are first **initialized** and subsequently loop over **Model Training**, **Data Acquisition** and **Data Annotation** steps for T iterations, or until a pre-defined performance on $\mathcal{D}_{\textbf{val}}$ is reached.

2.2 Active Learning Initialization

To initialize AL, the total number of AL iterations can be simply calculated by $T = \frac{b}{k}$, where b is the budget and k the acquisition size.[2] Then, a data initialization policy selects the first k data points from $\mathcal{D}_{\textbf{pool}}$ to be annotated and update the labeled dataset $\mathcal{D}_{\textbf{lab}}$. The most common approach to select the first batch of data for annotation is stratified random sampling (Gal et al., 2017).

2.3 Model Training

In the first step of the AL loop, a model $\mathcal{M}_i$ is trained with the available labeled data $\mathcal{D}_{\textbf{lab}}$ at iteration i. If $\mathcal{M}_i$ is a task-specific architecture (Shen et al., 2017; Siddhant and Lipton, 2018; Prabhu et al., 2019), it is simply trained from scratch on $\mathcal{D}_{\textbf{lab}}$ until convergence. If $\mathcal{M}_i$ is based on a pretrained LM (Yuan et al., 2020; Ein-Dor et al., 2020), then it is initialized with the pretrained weights and fine-tuned to the task on $\mathcal{D}_{\textbf{lab}}$ by adding a task-specific output classification layer and updating all model parameters until convergence. Note that

[2]If the budget b is a percentage of the number of unlabeled data points then $T = \frac{b|\mathcal{D}_{\textbf{pool}}|}{k}$.

at each iteration i, the model parameters are initialized randomly if $\mathcal{M}_i$ is trained from scratch or from the original pretrained LM, respectively. Warm-starting the model (i.e. initializing $\mathcal{M}_i$ with the parameters of $\mathcal{M}_{i-1}$) has been shown to hinder the model's generalization ability (Ash and Adams, 2020). The AL loop stops if performance of $\mathcal{M}_i$ on $\mathcal{D}_{\textbf{val}}$ is equal or higher that the goal.

2.4 Data Acquisition

In this step, we use the acquisition function a to select the k most informative unlabeled samples from $\mathcal{D}_{\textbf{pool}}$ for annotation. The acquisition function usually uses the trained model $\mathcal{M}_i$ to rank the candidate unlabeled data. This is called a *warm-start* approach and the acquisition function formally is $a(\mathcal{M}_i, \mathcal{D}_{\textbf{pool}}, k)$. A *cold-start* acquisition function typically does not use the model and selects data based on their input representations $a(\mathcal{D}_{\textbf{pool}}, k)$.

There are two main strategies for acquiring data: *uncertainty* and *diversity* sampling. Uncertainty sampling aims to select the most uncertain data based on the model's predictive uncertainty. The assumption is that the most uncertain data are the most difficult ones for the model, and therefore the most useful to facilitate training. Typical uncertainty-based acquisition functions include LEAST CONFIDENCE (Lewis and Gale, 1994) that sorts $\mathcal{D}_{pool}$ by the probability of *not* predicting the most confident class, in descending order, ENTROPY (Shannon, 1948) the selects samples that maximize the predictive entropy, and BALD (Houlsby et al., 2011), short for Bayesian Active Learning by Disagreement, that chooses data points that maximize the mutual information between predictions and model's posterior probabilities. BATCHBALD (Kirsch et al., 2019) is a recently introduced extension of BALD that *jointly* scores points by estimating the mutual information between multiple data points and the model parameters. This iterative algorithm aims to find *batches* of informative data points, in contrast to BALD that chooses points that are informative individually. Uncertainty sampling is a warm-start approach since it requires confidence scores from the trained model for all candidate unlabeled data.

On the other hand, diversity-based approaches aim to exploit the heterogeneity of the feature space and typically use clustering to choose a diverse set of points from $\mathcal{D}_{\textbf{pool}}$ (Wang and Ye, 2015; Sener and Savarese, 2018; Zeng et al., 2019). A diversity-based acquisition function can be either cold-start or warm-start. There are also hybrid approaches that aim to select data based on both uncertainty and diversity sampling (He et al., 2014; Yang et al., 2015; Erdmann et al., 2019; Yuan et al., 2020; Ash et al., 2020), and other methods that use reinforcement learning (Fang et al., 2017; Liu et al., 2018).

In our work, we use acquisition functions based on uncertainty sampling (§3.3), but any acquisition function that takes as input the unlabeled data, the acquisition size and, if applied, the model, and outputs a batch of k data points could be used, $\mathcal{Q}_i = a(\mathcal{M}_i, k, \mathcal{D}_{\textbf{pool}})$. Comparison of different types of acquisitions functions is out of the scope of this paper.

2.5 Data Annotation

Finally, the acquired set $\mathcal{Q}_i$ of k data points at iteration i is passed to an oracle for annotation. After acquiring labels, $\mathcal{Q}_i$ is appended to the labeled dataset $\mathcal{D}_{\textbf{lab}}$ and subsequently removed from $\mathcal{D}_{\textbf{pool}}$. The remaining budget b is adjusted accordingly. If it has been exhausted, AL stops. Otherwise iteration i+1 begins from the **Model Training** step with the updated $\mathcal{D}_{\textbf{lab}}$ and $\mathcal{D}_{\textbf{pool}}$ datasets.

We note that, following previous work (Siddhant and Lipton, 2018; Yuan et al., 2020; Ein-Dor et al., 2020), we use budget as a stopping criterion to facilitate fair comparison between the various methods considered. However, there are various AL stopping criteria for practitioners (Vlachos, 2008; Bloodgood and Vijay-Shanker, 2009) which are beyond the scope of this paper.

3 BALM: Bayesian Active Learning with Pretrained Language Models

Our aim is to improve LM-based AL to (1) account for varying data resource availability; (2) tackle the instability of LM fine-tuning; and (3) improve data acquisition with better calibrated confidence scores. For that purpose, we propose Bayesian Active Learning with pretrained language models (BALM) following the standard AL pipeline (§2).

In the AL initialization step (§2.2), we first adapt the LM using all the available unlabeled data of the downstream task (§3.1). We then propose a fine-tuning approach of the model (§3.2) that adjusts to all data availability settings (i.e. the low-resource setting at the first iterations, and the high-resource at the later iterations) during training (§2.2). Last, we extract uncertainty estimates from the adapted

Algorithm 1: BALM algorithm

Input: unlabeled data $\mathcal{D}_{\textbf{pool}}$, pretrained language model $\mathcal{P}(x; W_0)$, acquisition size k, AL iterations T, acquisition function a

1 $\mathcal{D}_{\textbf{lab}} \leftarrow \emptyset$
2 $\mathcal{P}_{\text{TAPT}}(x; W_0') \leftarrow$ Train $\mathcal{P}(x; W_0)$ on $\mathcal{D}_{\textbf{pool}}$
3 $\mathcal{Q}_0 \leftarrow \text{RANDOM}(.), |\mathcal{Q}_0| = k$
4 $\mathcal{D}_{\textbf{lab}} = \mathcal{D}_{\textbf{lab}} \cup \mathcal{Q}_0$
5 $\mathcal{D}_{\textbf{pool}} = \mathcal{D}_{\textbf{pool}} \setminus \mathcal{Q}_0$
6 **for** $i \leftarrow 1$ **to** T **do**
7 $\quad \mathcal{M}_i(x; [W_0', W_c]) \leftarrow$ Initialize from $\mathcal{P}_{\text{TAPT}}(x; W_0')$
8 $\quad \mathcal{M}_i(x; W_i) \leftarrow$ Train model on $\mathcal{D}_{\textbf{lab}}$
9 $\quad \mathcal{Q}_i \leftarrow a(\mathcal{M}_i, \mathcal{D}_{\textbf{pool}}, k)$
10 $\quad \mathcal{D}_{\textbf{lab}} = \mathcal{D}_{\textbf{lab}} \cup \mathcal{Q}_i$
11 $\quad \mathcal{D}_{\textbf{pool}} = \mathcal{D}_{\textbf{pool}} \setminus \mathcal{Q}_i$
12 **end**

Output: $\mathcal{D}_{\textbf{lab}}$

model using a probabilistic framework (§3.3) to improve uncertainty-based data acquisition.

In our experiments, we use BERT (Devlin et al., 2019), a state-of-the-art pretrained language model, as our AL classification model but our method is independent of the chosen LM.

3.1 LM Adaptation during AL Initialization

Inspired by recent work on transfer learning that shows improvements in downstream classification performance by continuing the pretraining of the LM with the task data (Howard and Ruder, 2018; Gururangan et al., 2020), we add an extra step in the AL initialization by continuing pretraining the LM. To this end, we use Task-Adaptive Pretraining (TAPT) to the AL setting. Formally, we use an LM, such as BERT (Devlin et al., 2019), $\mathcal{P}(x; W_0)$ with weights W_0, that has been already pretrained on a large corpus. We fine-tune $\mathcal{P}(x; W_0)$ with the available unlabeled data of the downstream task $\mathcal{D}_{\textbf{pool}}$, resulting in the task-adapted LM $\mathcal{P}_{\text{TAPT}}(x; W_0')$ with new weights W_0' (cf. line 2 of algorithm 1).

3.2 AL Classification Model Fine-tuning

We now use the adapted LM $\mathcal{P}_{\text{TAPT}}(x; W_0')$ for active learning. At each iteration i, we initialize our model $\mathcal{M}_i$ with the pretrained weights W_0' and we add a task-specific feedforward layer for classification W_c on top of the `[CLS]` token representation of BERT-based $\mathcal{P}_{\text{TAPT}}$. We fine-tune the classification model $\mathcal{M}_i(x; [W_0', W_c])$ with all $x \in \mathcal{D}_{\textbf{lab}}$. (cf. line 6 to 8 of algorithm 1).

Recent work in AL (Ein-Dor et al., 2020; Yuan et al., 2020) uses the standard fine-tuning method proposed in Devlin et al. (2019) which includes a fixed number of 3 training epochs, a learning rate between 2e-5 and 5e-5, learning rate warmup over the first 10% of the steps and AdamW optimizer (Loshchilov and Hutter, 2019) without bias correction, among other hyperparameters. We follow a different approach by taking into account insights from few-shot fine-tuning literature (Mosbach et al., 2021; Zhang et al., 2020) that proposes longer fine-tuning. We also follow Dodge et al. (2020) that demonstrates more robust BERT fine-tuning by increasing the number of evaluations steps during training.

We combine these guidelines to our fine-tuning approach by using early stopping with 20 epochs based on the validation loss, learning rate 2e-5, bias correction and 5 evaluation steps per epoch. However, increasing the number of epochs from 3 to 20, also increases the warmup steps (10% of total steps[3]) almost 7 times. This may be problematic in scenarios where the dataset is large but the optimal number of epochs may be small (e.g. 2 or 3). To account for this limitation in our AL setting where the size of training set changes at each iteration, we propose a simple empirical warmup approach by selecting the warmup steps as $min(10\%$ of total steps$, 100)$. We denote standard fine-tuning as SFT and our approach as FT+.

3.3 Uncertainty Estimation for Data Acquisition

After fine-tuning the classification model $\mathcal{M}_i$ with $\mathcal{D}_{\textbf{lab}}$, we use it to acquire uncertainty estimates for all candidate data points in $\mathcal{D}_{\textbf{pool}}$. We use uncertainty sampling by selecting the k most uncertain data from $\mathcal{D}_{\textbf{pool}}$ for annotation (cf. line 9 of algorithm 1). Instead of using the output softmax probabilities for each class, we use a probabilistic formulation of deep neural networks in order to acquire better calibrated scores.

Monte Carlo (MC) dropout (Gal and Ghahramani, 2016) is a simple yet effective method for performing approximate variational inference, based on dropout (Srivastava et al., 2014). Gal and Ghahramani (2016) prove that by simply perform-

[3]Some guidelines propose an even smaller number of warmup steps, such as 6% in RoBERTa (Liu et al., 2020).

DATASETS	TRAIN	VAL	TEST	k	C
TREC-6	4.9K	546	500	1%	6
DBPEDIA	20K	2K	70K	1%	14
IMDB	22.5K	2.5K	25K	1%	2
SST-2	60.6K	6.7K	871	1%	2
AGNEWS	114K	6K	7.6K	0.5%	4

Table 1: Datasets statistics for $\mathcal{D}_{\textbf{pool}}$, $\mathcal{D}_{\textbf{val}}$ and $\mathcal{D}_{\textbf{test}}$ respectively. k stands for the acquisition size (% of $\mathcal{D}_{\textbf{pool}}$) and C the number of classes.

ing *dropout during the forward pass in making predictions*, the output is equivalent to the prediction when the parameters are sampled from a variational distribution of the true posterior. Therefore, dropout during inference results into obtaining predictions from different parts of the network.

Our BERT-based $\mathcal{M}_i$ model uses dropout layers during training for regularization. We apply MC dropout by simply activating them during test time and we perform multiple stochastic forward passes. Formally, we do N passes of every $x \in \mathcal{D}_{\textbf{pool}}$ through $\mathcal{M}_i(x; W_i)$ to acquire N different output probability distributions for each x.

Four uncertainty acquisition functions are used in our work: LEAST CONFIDENCE, ENTROPY, BALD and BATCHBALD (§2.4). Note that LEAST CONFIDENCE, ENTROPY and BALD have been used in AL for NLP by Siddhant and Lipton (2018). To the best of our knowledge, BATCHBALD is evaluated for the first time in the NLP domain.

4 Experimental Setup

4.1 Tasks & Datasets

We experiment with five diverse natural language understanding tasks including binary and multi-class labels and varying dataset sizes (Table 1). The first task is question classification using the six-class version of the small TREC-6 dataset of open-domain, fact-based questions divided into broad semantic categories (Voorhees and Tice, 2000). We also evaluate our approach on sentiment analysis using the binary movie review IMDB dataset (Maas et al., 2011) and the binary version of the SST-2 dataset (Socher et al., 2013). We finally use the large-scale AGNEWS and DBPEDIA datasets from Zhang et al. (2015) for topic classification. We undersample the latter and form a $\mathcal{D}_{\textbf{pool}}$ of 20K examples and $\mathcal{D}_{\textbf{val}}$ 2K.

4.2 Training & AL Details

We use BERT-BASE (Devlin et al., 2019) and fine-tune it (TAPT §3.1) for 100K steps, with learning rate 2e-05 and the rest of hyperparameters as in Gururangan et al. (2020) using the HuggingFace library (Wolf et al., 2020). We evaluate the model 5 times per epoch on $\mathcal{D}_{val}$ and keep the one with the lowest validation loss as in Dodge et al. (2020). We use the code provided by Kirsch et al. (2019) for the uncertainty-based acquisition functions and Yuan et al. (2020) for ALPS, BADGE and BERTKM. We use the standard splits provided for all datasets, if available, otherwise we randomly sample a validation set. We test all models on a held-out test set. We repeat all experiments with five different random seeds resulting into different initializations of $\mathcal{D}_{\textbf{lab}}$ and the weights of the extra task-specific output feedforward layer. For all datasets we use as budget the 15% of $\mathcal{D}_{\textbf{pool}}$. Each experiment is run on a single Nvidia Tesla V100 GPU. More details are provided in the Appendix A.1.

4.3 Baselines

Acquisition functions We compare uncertainty sampling (§3.3) with four baseline acquisition functions. The first is the standard AL baseline, **RANDOM**, which applies uniform sampling and selects k data points from $\mathcal{D}_{\textbf{pool}}$ at each iteration. The second is **BADGE** (Ash et al., 2020), an acquisition function that aims to combine diversity and uncertainty sampling. The algorithm computes *gradient embeddings* g_x for every candidate data point x in $\mathcal{D}_{\textbf{pool}}$ and then uses clustering to select a batch. Each g_x is computed as the gradient of the cross-entropy loss with respect to the parameters of the model's last layer. We also compare against a recently introduced cold-start acquisition function called **ALPS** (Yuan et al., 2020). ALPS acquisition uses the masked language model (MLM) loss of BERT as a proxy for model uncertainty in the downstream classification task. Specifically, aiming to leverage both uncertainty and diversity, ALPS forms a *surprisal embedding* s_x for each x, by passing the unmasked input x through the BERT MLM head to compute the cross-entropy loss for a random 15% subsample of tokens against the target labels. ALPS clusters these embeddings to sample k sentences for each AL iteration. Last, following Yuan et al. (2020), we use **BERTKM** as a diversity baseline, where the l_2 normalized BERT output embeddings are used for clustering.

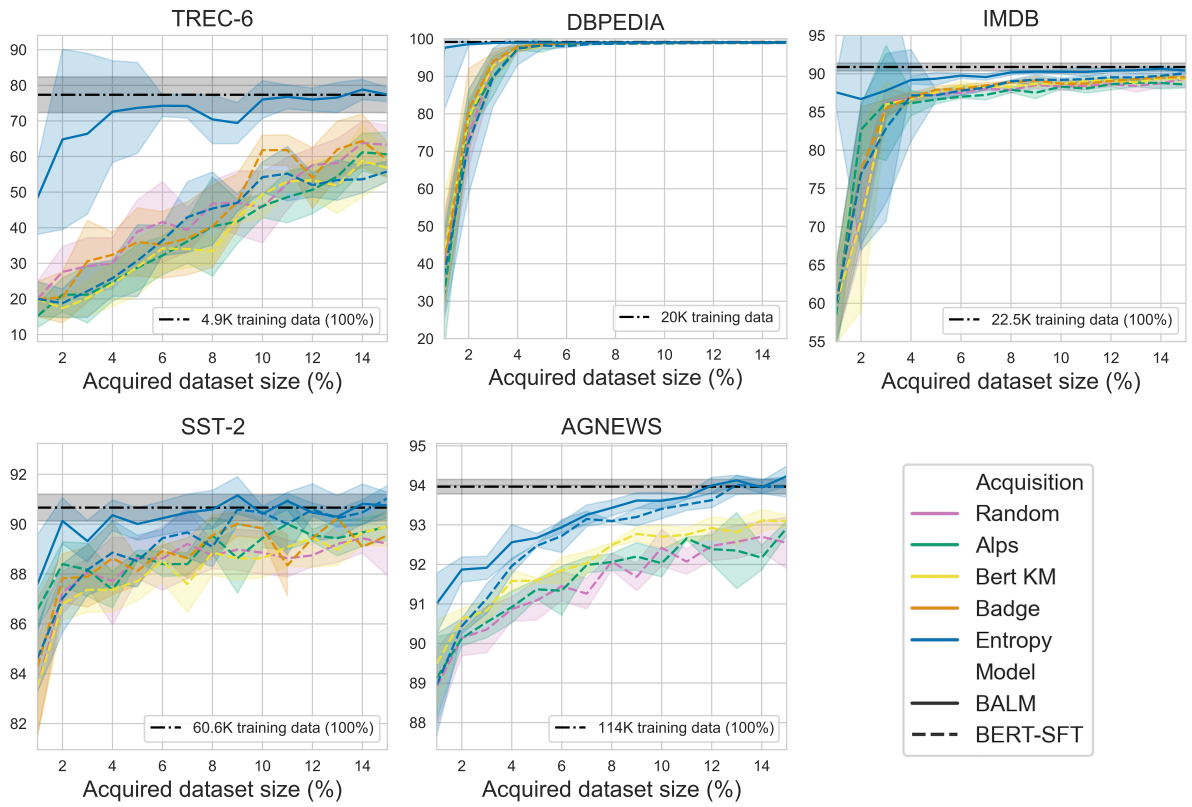

Figure 1: Test accuracy during AL iterations using BALM with ENTROPY against RANDOM, ALPS, BADGE and BERTKM acquisition functions. The dotted line denotes experiments with BERT and standard fine-tuning (SFT) and the solid line with BERT-TAPT and FT+. We plot the median and standard deviation across five runs.

Models & Fine-tuning Methods We also evaluate (§6) two variants of the pretrained language model; the original **BERT** model, used in Yuan et al. (2020) and Ein-Dor et al. (2020)[4], and our adapted model **BERT-TAPT** (§3.1), and two fine-tuning methods; our proposed fine-tuning approach **FT+** (§3.2) and standard BERT fine-tuning **SFT**.

5 Results

Figure 1 presents the results for all datasets [5]. Our proposed method BALM consists of the BERT-TAPT model (§3.1), FT+ fine-tuning method (§3.2) and ENTROPY acquisition (§3.3). For *all* experiments with ENTROPY acquisition, we use MC dropout with $N = 5$. We show that BALM consistently outperforms all baselines across datasets.

Data Efficiency We first observe that BALM achieves large data efficiency since it reaches the full-dataset performance within the 15% budget for all datasets. The performance of BALM is mostly notable in the smaller datasets. In TREC-6, it achieves the goal accuracy with almost 10% annotated data, while in DBPEDIA only in the first iteration with 2% of the data. In the first AL iteration in IMDB, BALM results only in 2.5 points of accuracy lower than the performance equivalent to using 100% of the data, which it later achieves after acquiring 15% of the data. In the larger SST-2 and AGNEWS datasets, BALM is closer to the baselines but still outperforms them, achieving the full-dataset performance with 8% and 12% of the data respectively.

Training Strategy We also observe that in all datasets, the addition of our proposed pretraining step (TAPT § 3.1) and fine-tuning technique (FT+ 3.2) leads to large performance gains, especially in the first AL iterations. This is particularly evident in TREC-6, DBPEDIA and IMDB datasets, where after the *first* AL iteration (i.e. equivalent to 2% of training data) BALM with ENTROPY is 45, 30 and 12 points in accuracy, respectively, higher than the ENTROPY baseline with BERT and SFT. This

[4] Ein-Dor et al. (2020) evaluate various acquisition functions, including entropy with MC dropout, and use BERT with the standard fine-tuning approach (SFT).

[5] We do not evaluate BADGE on AGNEWS because of the increased time complexity of the algorithm: $\mathcal{O}(Cknd)$ for a C-way classification task, k queries, n points in $\mathcal{D}_{\textbf{pool}}$, and d-dimensional BERT embeddings.

is a rather interesting finding, since our simple additions in the training strategy of the model proved to be particularly effective and resulting in large performance improvements.

Acquisition Strategy We finally observe that the performance curves of the various acquisition functions considered (i.e. dotted lines) are generally close to each other, suggesting that the choice of the acquisition strategy does not affect substantially the AL performance. In other words, we conclude that *the training strategy is more important than the acquisition strategy*. We find that uncertainty sampling with ENTROPY is generally the best performing acquisition function, followed by BADGE. Still, finding a universally well-performing acquisition function, independent of the training strategy, is an open research question. Our findings show that uncertainty sampling is the strongest approach, with room for improvement over the competitive random sampling baseline (§6).

6 Analysis & Discussion

Task-Adaptive Pretraining We present details of TAPT (§3.1) and reflect on its effectiveness in the AL pipeline. Following Gururangan et al. (2020), we continue pretraining BERT for the MLM task using all the unlabeled data $\mathcal{D}_{\textbf{pool}}$ for all datasets separately. We plot the learning curves of BERT-TAPT for all datasets in Figure 2. We first observe that the masked LM loss is steadily decreasing for DBPEDIA, IMDB and AGNEWS across optimization steps, which correlates with the high early AL performance gains of TAPT in these datasets (Fig. 1). We also observe that the LM overfits in TREC-6 and SST-2 datasets. We attribute this to the very small training dataset of TREC-6 and the informal textual style of SST-2. Although SST-2 includes approximately 67K of training data, the sentences are very short (i.e. average length of 9.4 words per input sentence). We hypothesize the LM overfits because of the lack of long and diverse sentences. More details on TAPT can be found in the Appendix A.2.

Few-shot Fine-tuning We highlight the importance of considering the few-shot learning problem in the AL pipeline during the first iterations which is often neglected in literature. This is more important when using pretrained LMs, since they are overparameterized models that require adapting the training scheme when low data resources are available to ensure robustness.

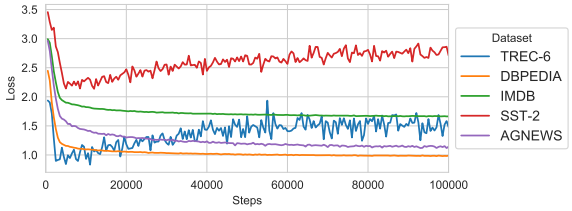

Figure 2: Validation MLM loss during TAPT.

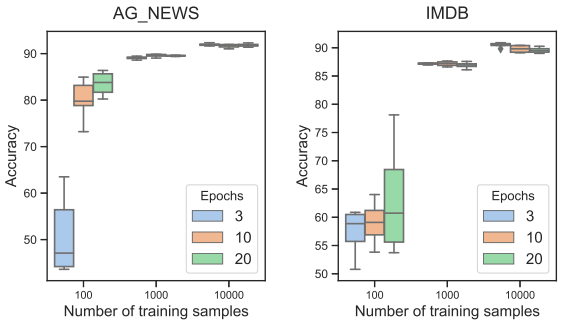

Figure 3: Few-shot standard BERT fine-tuning.

To illustrate the inefficiency of standard fine-tuning (SFT), we randomly undersample AGNEWS and IMDB to form low, medium and high data settings (i.e. 100, 1,000 and 10,000 training samples) and train BERT for a fixed number of 3, 10, and 20 epochs. Figure 3 shows that SFT is suboptimal for low data settings, indicating that more optimization steps are needed for the model to adapt to the few training samples (Mosbach et al., 2021; Zhang et al., 2020). As the training samples increase fewer epochs are often better. It is thus evident that there is not a clearly optimal way to choose a predefined number of epochs to train the model given the number of training examples. This motivates the need to find a fine-tuning policy for AL that efficiently adapts to the data resource setting of each iteration (independent of the number of training examples or dataset), which is mainly tackled by our proposed fine-tuning approach FT+ (§3.2).

Ablation Study We also conduct an ablation study to show that our proposed AL training methods, (i) the pretraining step (TAPT §3.1) and (ii) the fine-tuning method (FT+ §3.2), provide large gains compare to standard BERT fine-tuning (SFT) in terms of accuracy, data efficiency and uncertainty calibration. We therefore compare BERT with SFT, BERT with FT+ and BERT-TAPT with FT+ (BALM). Along with test accuracy, we also evaluate each AL model on a benchmark of uncertainty estimation metrics as proposed by Ovadia et al. (2019),

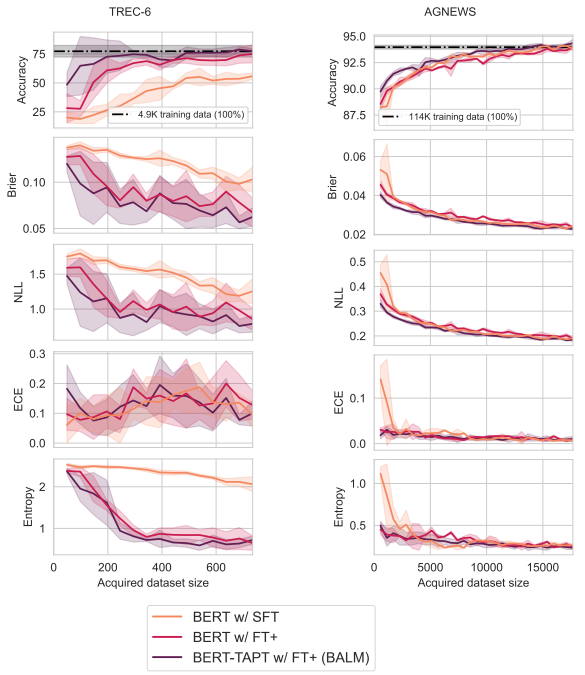

Figure 4: BALM ablation study.

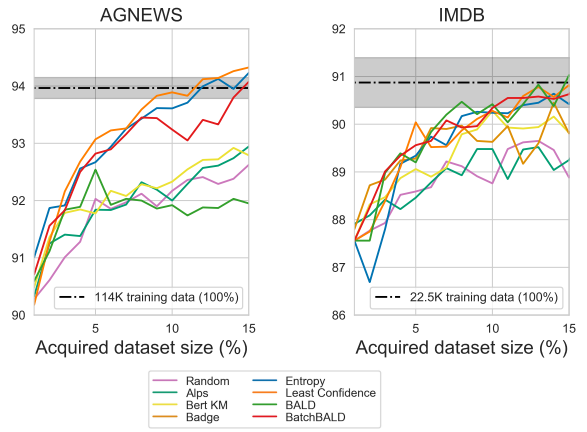

Figure 5: Comparison of acquisition functions using TAPT and FT+ in training BERT.

namely Brier score, negative log likelihood (NLL), expected calibration error (ECE) and entropy. A well-calibrated model should have high accuracy and low values on the uncertainty metrics.

Figure 4 shows the results for the smallest and largest datasets, TREC-6 and AGNEWS respectively. For TREC-6, training BERT with our fine-tuning approach FT+ provides large gains both in accuracy and uncertainty calibration, showing how important it is to fine-tune the LM for a larger number of epochs in low resource settings. For the larger dataset, AGNEWS, we see that BERT with SFT performs equally to FT+ which is the ideal scenario. We see that our fine-tuning approach does not deteriorate the performance of BERT because of the large increase in warmup steps (see §3.2), showing that our simple fix provides robust results in both high and low resource settings.

After demonstrating that FT+ yields better results than SFT, we next compare BALM against BERT with FT+. We observe that in both datasets BERT-TAPT outperforms BERT, with this being particularly evident in the early iterations. This finding confirms our hypothesis that by implicitly using the entire pool of unlabeled data in the extra pretraining step (TAPT), we boost the performance of the AL classification model using less data.

Performance of Acquisition Functions In our BALM experiments so far, we showed results with ENTROPY. We have also experimented with various uncertainty-based acquisition functions, i.e. LEAST CONFIDENCE, BALD and BATCHBALD (§3.3), and our findings show that all functions provide similar performance, except for BALD that slightly underperforms. This makes our approach agnostic to the selected uncertainty-based acquisition method. We also evaluate our proposed methods with our baseline acquisition functions, i.e. RANDOM, ALPS, BERTKM and BADGE, since our training strategy is orthogonal to the acquisition strategy. We compare all acquisition functions with BALM for AGNEWS and IMDB in Figure 5. We observe that in general uncertainty-based acquisition performs better compared to diversity, while all acquisition strategies have benefited from our BALM training strategy (TAPT and FT+). We discuss the efficiency of the methods in the Appendix A.3.

7 Conclusions & Future Work

We have presented Bayesian Active Learning with pretrained language Models (BALM) consisting of (i) an extra pretraining step with the unlabeled task specific data, (ii) a simple yet effective fine-tuning method for the downstream model and (iii) use of MC dropout to acquire well-calibrated confidence scores for uncertainty sampling. BALM accounts for the few-shot learning phase of AL while still adapts effectively to the high-resource setting of the last iterations. Our findings also show that the proposed training strategy is more effective in improving AL performance that the selected acquisition function. In the future, we aim to investigate semi-supervised learning methods to leverage unlabeled data during training the downstream model.

References

Jordan T. Ash and Ryan P. Adams. 2020. On warm-starting neural network training.

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*.

Daniel Beck, Lucia Specia, and Trevor Cohn. 2013. Reducing annotation effort for quality estimation via active learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 543–548.

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

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4(1):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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *ArXiv*.

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 theConference on Empirical Methods in Natural Language Processing*, pages 7949–7962.

Sean P. Engelson and Ido Dagan. 1996. Minimizing manual annotation cost in supervised training from corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 319–326.

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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2223–2234.

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 International Conference on Machine Learning*, volume 48, pages 1050–1059.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep Bayesian active learning with image data. In *Proceedings of the International Conference on Machine Learning*, volume 70, pages 1183–1192.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks.

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, Online. Association for Computational Linguistics.

Gholamreza Haffari, Maxim Roy, and Anoop Sarkar. 2009. Active learning for statistical phrase-based machine translation. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 415–423.

Tianxu He, Shukui Zhang, Jie Xin, Pengpeng Zhao, Jian Wu, Xuefeng Xian, Chunhua Li, and Zhiming Cui. 2014. An active learning approach with uncertainty, representativeness, and diversity. *Scientific WorldJ ournal*, 2014:827586.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. 2011. Bayesian active learning for classification and preference learning. *ArXiv*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 328–339.

Fariz Ikhwantri, Samuel Louvan, Kemal Kurniawan, Bagas Abisena, Valdi Rachman, Alfan Farizki Wicaksono, and Rahmad Mahendra. 2018. Multi-task active learning for neural semantic role labeling on low resource conversational corpus. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 43–50.

Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource deep entity resolution with transfer and active learning. In *Proceedings of the Conference of the Association for Computational Linguistic*, pages 5851–5861.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. 2019. BatchBALD: Efficient and diverse batch acquisition for deep bayesian active learning. In *Neural Information Processing Systems*, pages 7026–7037.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *In Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Shoushan Li, Shengfeng Ju, Guodong Zhou, and Xiaojun Li. 2012. Active learning for imbalanced sentiment classification. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 139–148.

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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{bert}a: A robustly optimized {bert} pretraining approach.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

David Lowell and Zachary C Lipton. 2019. Practical obstacles to deploying active learning. *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, pages 21–30.

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 *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. On the stability of fine-tuning {bert}: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. 2019. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, volume 32, pages 13991–14002.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Ameya Prabhu, Charles Dognin, and Maneesh Singh. 2019. Sampling bias in deep active classification: An empirical study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, pages 4056–4066.

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

Burr Settles. 2009. Active learning literature survey. Computer sciences technical report.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. In *Proceedings of the Workshop on Representation Learning for NLP*, pages 252–256.

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 Conference on Empirical Methods in Natural Language Processing*, pages 2904–2909.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

N Srivastava, G Hinton, A Krizhevsky, and others. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *Annual Meeting of the Association for Computational Linguistics*.

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

Ellen Voorhees and Dawn Tice. 2000. The trec-8 question answering track evaluation. *Proceedings of the Text Retrieval Conference*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Zheng Wang and Jieping Ye. 2015. Querying discriminative and representative samples for batch mode active learning. *ACM Transactions on Knowledge Discovery from Data*, 9(3).

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

Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G Hauptmann. 2015. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113(2):113–127.

Michelle Yuan, Hsuan-Tien Lin, and Jordan Boyd-Graber. 2020. Cold-start active learning through self-supervised language modeling.

Xiangkai Zeng, Sarthak Garg, Rajen Chatterjee, Udhyakumar Nallasamy, and Matthias Paulik. 2019. Empirical evaluation of active learning techniques for neural MT. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 84–93.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning. *ArXiv*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, pages 649–657. Curran Associates, Inc.

A Appendix

A.1 Hyperparameters & Dataset Details

In this section we provide details of all the datasets we used in this work and the hyperparparameters used for training the model. For TREC-6, IMDB and SST-2 we randomly sample 10% from the training set to serve as the validation set, while for AGNEWS we sample 5%. For the DBPEDIA dataset we undersample both training and validation datasets (from the standard splits) to facilitate our AL simulation (i.e. the original dataset consists of 560K training and 28K validation data examples). For all datasets we use the standard test set, apart from the SST-2 dataset that is taken from the GLUE benchmark (Wang et al., 2019) we use the development set as the held-out test set.

For all datasets we train BERT-BASE (Devlin et al., 2019) from the HuggingFace library (Wolf et al., 2020) in Pytorch (Paszke et al., 2019). We train all models with batch size 16, learning rate $2e-5$, no weight decay, AdamW optimizer with epsilon $1e-8$. For all datasets we use maximum sequenxe length of 128, except for IMDB and AGNEWS that contain longer input texts, where we use 256. To ensure reproducibility and fair comparison between the various methods under evaluation, we run all experiments with the same five seeds that we randomly selected from the range $[1, 9999]$.

MODEL	TREC-6	DBPEDIA	IMDB	SST-2	AGNEWS
VALIDATION SET					
BERT	94.4	99.1	90.7	93.7	94.4
BERT-TAPT	95.2	99.2	91.9	94.3	94.5
TEST SET					
BERT	80.6	99.2	91.0	90.6	94.0
BERT-TAPT	77.2	99.2	91.9	90.8	94.2

Table 2: Accuracy with 100% of data over five runs (different random seeds).

A.2 Task-Adaptive Pretraining (TAPT) & Full-Dataset Performance

As discussed in §3.1 and §6, we continue training the BERT-BASE (Devlin et al., 2019) pretrained masked language model using the available data $\mathcal{D}_{\textbf{pool}}$. We explored various learning rates between 1e-4 and 1e-5 and found the latter to produce the lowest validation loss. We trained each model (one for each dataset) for up to 100K optimization steps, we evaluated on $\mathcal{D}_{\textbf{val}}$ every 500 steps and saved the checkpoint with the lowest validation loss. We used the resulting model in our BALM experiments.

	TREC-6	SST-2	IMDB	DBPEDIA	AGNEWS
RANDOM	0/0	0/0	0/0	0/0	0/0
ALPS	0/57	0/478	0/206	0/134	0/634
BADGE	0/63	0/23110	0/1059	0/192	-
BERTKM	0/47	0/2297	0/324	0/137	0/3651
ENTROPY	81/0	989/0	557/0	264/0	2911/0
LEAST CONFIDENCE	69/0	865/0	522/0	256/0	2607/0
BALD	69/0	797/0	524/0	256/0	2589/0
BATCHBALD	69/21	841/1141	450/104	256/482	2844/5611

Table 3: Runtimes (in seconds) for all datasets. In each cell of the table we present a tuple i/s where i is the *inference time* and s the *selection time*. *Inference time* is the time for the model to perform a forward pass for all the unlabeled data in $\mathcal{D}_{\textbf{pool}}$ and *selection time* is the time that each acquisition function requires to rank all candidate data points and select k for annotation (for a single iteration). Since we cannot report the runtimes for every model in the AL pipeline (at each iteration the size of $\mathcal{D}_{\textbf{pool}}$ changes), we provide the median.

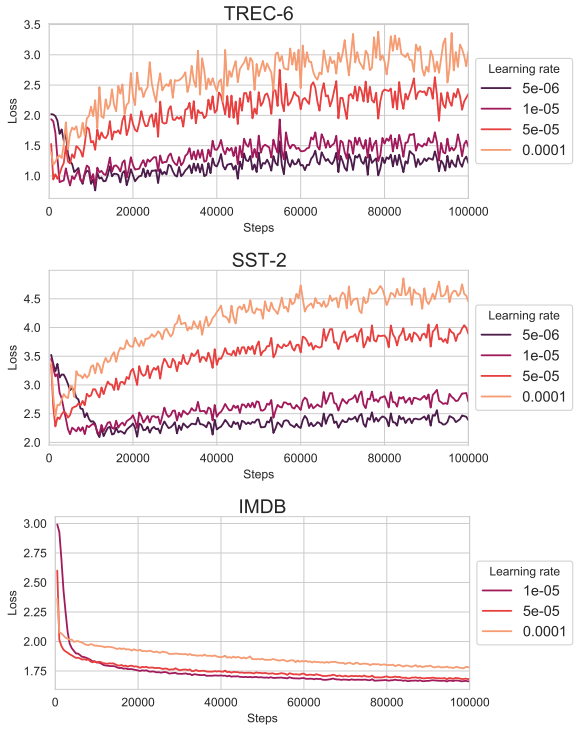

Figure 6: Learning curves of TAPT for various learning rates.

We plot the learning curves of masked language modeling task (TAPT) for three datasets and all considered learning rates in Figure 6. We notice that a smaller learning rate facilitates the training of the MLM.

In Table 2 we provide the validation and test accuracy of BERT and BERT-TAPT for all datasets. We present the mean across runs with three random seeds. For fine-tuning the models, we used the proposed approach FT+ (§3.2).

A.3 Efficiency of Acquisition Functions

In this section we discuss the efficiency of the eight acquisition functions considered in this work; RANDOM, ALPS, BADGE, BERTKM, ENTROPY, LEAST CONFIDENCE, BALD and BATCHBALD.

In Table 3 we provide the runtimes for all acquisition functions and datasets. Each AL experiments consists of multiple iterations and (therefore multiple models), each with a different training dataset $\mathcal{D}_{\textbf{lab}}$ and pool of unlabeled data $\mathcal{D}_{\textbf{pool}}$. In order to evaluate how computationally heavy is each method, we provide the *median* of all the models in one AL experiment. We calculate the runtime of two types of functionalities. The first is the *inference time* and stands for the forward pass of each $x \in \mathcal{D}_{\textbf{pool}}$ to acquire confidence scores for uncertainty sampling. RANDOM, ALPS, BADGE and BERTKM do not require this step so it is only applied of uncertainty-based acquisition where acquiring uncertainty estimates with MC dropout is needed. The second functionality is *selection time* and measures how much time each acquisition function requires to rank and select the k data points from $\mathcal{D}_{\textbf{pool}}$ to be labeled in the next step of the AL pipeline. RANDOM, ENTROPY, LEAST CONFIDENCE and BALD perform simple equations to rank the data points and therefore so do not require selection time. On the other hand, ALPS, BADGE, BERTKM and BATCHBALD perform iterative algorithms that increase selection time. From all acquisition functions ALPS and BERTKM are faster because they do not require the inference step of all the unlabeled data to the model. ENTROPY, LEAST CONFIDENCE and BALD require the same

time for selecting data, which is equivalent for the time needed to perform one forward pass of the entire $\mathcal{D}_{\textbf{pool}}$. Finally BADGE and BATCHBALD are the most computationally heavy approaches, since both algorithms require multiple computations for the *selection time*. RANDOM has a total runtime of zero seconds, as expected.