2021 International Conference on Data Mining Workshops (ICDMW) | 978-1-6654-2427-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICDMW53433.2021.00050

NimbleLearn: A Scalable and Fast Batch-mode Active Learning Approach

1st Ruoyan Kong
University of Minnesota
Minneapolis, USA
kong0135@umn.edu

2nd Zhanlong Qiu
Amazon
Seattle, USA
qiuzhanl@amazon.com

3rd Yang Liu
Amazon
Seattle, USA
lyangl@amazon.com

4th Qi Zhao
Amazon
Seattle, USA
qqzhao@amazon.com

***Abstract*—Batch-mode active learning iteratively selects a batch of unlabeled samples for labelling to maximize model performance and reduce total runtime. To select the most informative and diverse batch, existing methods usually calculate the correlation between samples within a batch, leading to combinatorial optimization problems which are inefficient, complex, and limited to linear models for approximated solutions. In this paper, we propose NimbleLearn, a scalable deep imitation batch-mode active learning approach to address these drawbacks. NimbleLearn sequentially predicts an "ideal sample" by a deep policy network for each batch. Such ideal sample maximizes the model performance when combined with the labeled samples and the already-selected samples in the current batch. Unlike the existing batch-mode active learning methods which directly select one batch of samples from unlabeled ones, NimbleLearn reduces the dimension of the policy network output to the number of features (assuming the number of unlabeled samples is much greater than the number of features). In addition, NimbleLearn is a general framework and can be applied in both linear and nonlinear models. Experiments conducted on 4 public datasets show NimbleLearn can achieve similar or better performance as existing SOTA algorithms, while reducing the number of labeled samples and runtime by over 50%.**

***Index Terms*—active learning, batch-mode, imitation learning**

I. Introduction

Supervised learning needs enough labeled data to achieve a reasonable performance in practice. To label the data is usually costly and time-consuming. In order to reduce the cost of labeling while achieving a decent model performance, machine learning practitioners have been widely using active learning to train supervised models. In active learning, an agent selects a subset of unlabelled samples for labelling, and then a supervised model or classifier is retrained with those newly added labels together with the already labeled samples. This process is iterated until the model achieves a decent performance or the labelling budget is met. Active learning usually requires only a small number of labelled samples, while generating a considerable performance boost in practice [1]. Traditional active learning, the single-point active learning method, selects one sample per iteration for labeling, which is not time efficient since the model needs to be retrained for every labeled sample. On the other hand, batch-mode active learning [2]–[4] was proposed to select a batch of unlabeled samples in parallel for labeling, which reduces the training time (and labelling time). Since we will discuss the training of the active learning agent later, we will refer to the supervised model or classifier as the base model in this paper to avoid confusion from now on.

The key problem in batch-mode active learning is to design or train an agent to select the most informative batches with minimum redundancy in each batch, that is, $\pi(D^{lab}, D^{unl}, m) \longrightarrow \{x_1, x_2, ..., x_Q\}$, where π denotes the agent, D^{lab} and D^{unl} are labeled and unlabeled samples, m the base model, x_i one of selected samples and Q is the batch size (see Table II). There are $\binom{|D^{unl}|}{Q}$ ways to select a batch. Once the size of unlabelled samples D^{unl} becomes large (which is true in practice), scalability becomes an issue and overall speed is compromised. Therefore, most of these proposed algorithms are compromised with approximate solutions using linear base models [5], [6], or designing a policy heuristically [7]. However, those proposed algorithms are hard to extend to scenarios where we have a large size of unlabelled samples, or too slow.

We propose NimbleLearn, a scalable and fast batch-mode active learning approach. NimbleLearn trains an agent π to iteratively select one "ideal sample" for Q times in each batch. The "ideal sample" is expected to maximize the base model performance when combined with selected samples in the current batch and the labeled samples in a greedy fashion, namely $\pi(D^{lab}, D^{sel}, D^{unl}, m) \longrightarrow \{x_i\}$, where $D^{sel} = \{x_1, ..., x_{i-1}\}$ are selected samples of the current batch. Since the agent is implemented as a policy network in NimbleLearn, we will use these two terms exchangeably in this paper. NimbleLearn enables us to train a policy network which scales linearly with the number of features instead of the number of unlabelled samples to overcome the scalability issues. Also, NimbleLearn uses deep imitation learning to train π on a fully labelled source dataset, and then applies the trained π to select a batch of samples in an unlabeled target dataset. This actually is "active transfer" learning. Overall, this imitation learning strategy can learn an efficient policy and make NimbleLearn one of the fastest algorithms.

We conduct experiments on four benchmark datasets (balanced and imbalanced) for classification. Figure 1 shows the runtime of applying NimbleLearn versus the baselines against F1-score on Hate Speech classification. Results show that NimbleLearn requires fewer labeled samples and converges

978-1-6654-2427-1/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDMW53433.2021.00050

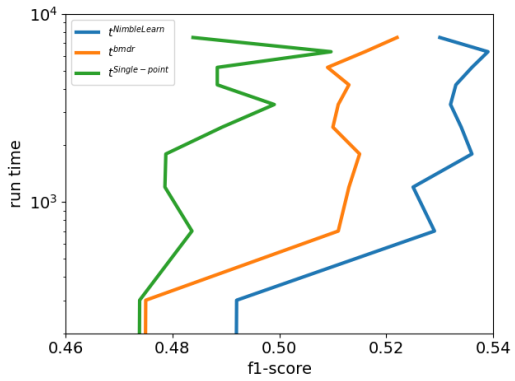


Fig. 1: Average runtime of single-point strategy, one of the tested batch-mode active learning baselines with the highest performance (BMDR), and NimbleLearn against base model F1-score on hate speech classification task.

TABLE I: NimbleLearn versus Batch-model Baselines.

	Uncer-tainty	Diver-sity	Ran-dom	BMDR	SPAL	K-Center	Nimble-Learn
Scalability	✓	✓	✓			✓	✓
Fast Algorithm				✓	✓	✓	✓
High Performance	✓	✓		✓			✓

faster compared with other tested batch-mode active learning algorithms (see Appendix A). We summarize the advantages of each algorithm in Table I, and key takeaways of NimbleLearn are listed as follows:

1) Scalability: NimbleLearn scales linearly with the size the of the features instead of the number of unlabelled samples, which well suits problems with large datasets.
2) Fast algorithm: NimbleLearn is more than 2x faster than tested baselines to reach similar performance due to the efficient selection algorithm.
3) High performance: NimbeLearn achieves the SOTA performance on various datasets, and even better performance on imbalanced datasets.

II. Related Work

A. Single-point Active Learning

Single-point active learning algorithms select one sample per round for labelling [2]. Different single-point active learning algorithms are distinguishable based on the method they use to select one sample per round. Early studies select the sample based on some informativeness of heuristics. Lewis and Gale [8] proposed an uncertainty based approach which selects one sample with the base model probability score closest to 0.5. Freund et al. [9] proposed the query-by-committee algorithm, which builds a committee of base models. In this case, the sample with the highest disagreement for prediction from the committee would be selected. Tong and Koller proposed an algorithm to select the sample that is closest to the decision boundary of the base model [10].

Later on, some algorithms considered the representativeness of the selected samples, and used the information of the unlabeled samples. McCallum and Nigam proposed to maximize the expected performance of the base model on unlabeled samples [11]. Muslea et al. [12] employed a semi-supervised learning approach to build a multi-view active learning algorithm. Despite these advancements, single-point active learning is inherently slow in application when a parallel labeling system is available, where multiple samples could be assigned to multiple human investigators simultaneously [3].

B. Batch-Mode Active Learning

Batch-mode active learning selects multiple samples each round to improve the time efficiency when multiple human investigators are available to label the samples in parallel [4]. Comparing with single-point active learning, batch-mode active learning has an added complexity to consider correlations between samples within a batch. Some methods use heuristics to approximate correlations, where researchers proposed a "diversity score" (the dissimilarity between one candidate sample and the samples already in the batch) and then added it to the "informativeness score" (which is used in the single-point active learning algorithms) with some preset weights [7], [13]. Xu et al. [14] and Li [15] et al. used the idea of semi-supervised clustering to measure the "diversity". Hoi et al. [16] considered the Fisher information score as the "informativeness" and selected the batch that maximizes it. However, because of pre-defined heuristics and preset weights, there is no guarantee to reach optimal base model performance from these proposed algorithms.

Besides using heuristics, some batch-mode active learning algorithms try to optimize the objectiveness around "informativeness" and "representativeness" of the selected batch. Wange and Ye [6] followed the idea of empirical risk minimization and tried to find the batch that maximizes the base model performance. This optimization is NP-hard and only has approximate solutions for choosing logistic regression as the base model. Huang et al. [17] proposed a min-max optimization strategy on "informativeness" and "representativeness" scores and provided solutions for the base model of SVM classifier using the quadratic loss function. Chakraborty et al. [18] proposed to optimize the maximum generalization capability of the selected batch. The original problem is NP-hard, and they further approximated it with an integer LP problem. Chattopadhyay [19] et al. tried to minimize the difference between the selected batch and the unlabeled samples, where the proposed objective is still an NP-hard integer programming problem. Fu et al. [20] proposed an approximated error reduction criterion which estimates the impact of one sample on other data points, and is implemented on hierarchical anchor graphs as the base model.

C. Active Learning as a Learning Process

Recent advancements fit iterative selection and labelling process into a reinforcement learning framework. Liu et al. [21] used a reinforcement learning framework to minimize

human annotation efforts while maximizing the base model performance. Lopes et al. [22] enabled the active learning agent to query the demonstrator for samples in specific states. However, these methods require many rounds to train a decent active learning agent, which contradicts the idea of applying batch-mode active learning to reduce overall runtime.

In high level, these algorithms train an active learning agent, which is perceived as a policy network to select the next unlabelled sample [23]. Supposing we have an unlabeled sample pool $D^{unl} = \{x_i\}_{i\in\{1,..,|D^{unl}|\}}$ and labeled sample pool $D^{lab} = \{(x_i, y_i)\}_{i\in\{1,..,|D^{lab}|\}}$, where x_i denotes feature vector of one sample and y_i is the label of sample class (see Table II). At each round r, the active learning agent selects an unlabeled sample based on an action probability vector $P_r = \pi(s_r)$, where P_r is an $|D^{unl}| \times 1$ vector, the ith element of P_r is the probability of selecting the ith unlabeled sample and s_r is the current training states. Here the s_r usually includes the feature vector of each unlabeled sample, and other information about labeled samples and the base model. The action A_r is to select the sample from D^{unl} with the highest probability:

$$A_r = argmax_{i\in\{1,...,|D^{unl}|\}} \{P_{r,i}\} \tag{1}$$

These reinforcement learning based approaches can apply to any base models. However, in their designs of policy networks, the input and output size scales with size of unlabelled samples. This can be problematic as the unlabelled sample size can be huge in practice.

D. Gap

Many existing algorithms are not scalable and quite slow, given that they introduce a complex combinatorial optimization problem that is NP-hard or require too many rounds of selection. Among these algorithms, some have approximate solutions for a specific linear base model. These drawbacks block the applications on selecting from large unlabelled datasets. Our proposed NimbleLearn can bridge the gap.

III. Method

A. Active Learning Agent as An Action Policy

In this section, we explain how to design a new policy network to represent the active learning agent while avoiding $O(|D^{unl}|)$ complexity. Suppose we budget for selecting Q samples per round for R rounds. At round r, to select the q-th sample, instead of outputting the probability of selecting each unlabeled sample, we output an $n\_feature \times 1$ "ideal sample" $P_{r,q}$. This "ideal sample" can improve the base model performance at most when adding it to the combined set of already-selected ones in current batch D^{sel} plus the labeled samples D^{lab}, in the retraining process. $n\_feature$ is the dimension of the feature vector. By this design, we decrease the size of the policy's output from $|D^{unl}|$ to $n\_feature$. Then the action $A_{r,q}$ is to select the sample from D^{unl} which is closest to this ideal sample:

$$A_{r,q} = argmin_{x\in D^{unl}}\ dist(x, P_{r,q}) \tag{2}$$

where $dist$ is any metric used to measure the distance between these 2 vectors ($0 \leq dist \leq 1$), e.g., 1 - cosine similarity.

The output of the policy network denotes a feature vector for an ideal sample. It takes state as inputs, which is the snapshot of the current sample states and the trained base model. Specifically, it is designed to include descriptive information of the current feature vectors (of labeled, unlabeled, and selected samples) and other general model information (e.g., percentage of each class in the labeled samples, model coefficients, entropy). The descriptive information of the current feature vectors could be the $[0, 1, ..., m] * \frac{100}{m}$th percentile of each dimension of the feature vectors of labeled/selected/unlabeled samples, see figure 2. With this design, we limit the policy network input size in $O(n\_feature)$.

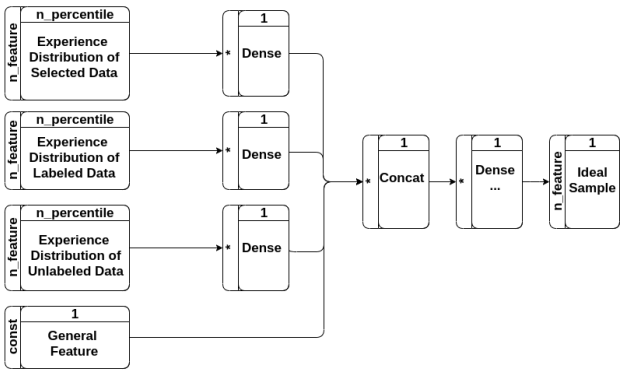


Fig. 2: The policy network for selecting an unlabeled sample to be added to current batch (selected samples). It takes the percentiles of each dimension of the feature vectors of labeled/selected/unlabeled samples and some general features of the training states as inputs, then goes through several dense layers, and outputs an ideal sample. * denotes an arbitrary number of hidden nodes.

B. Train Active Learning Agent via Imitation Learning

In this section, we explain how to learn the aforementioned policy. As discussed in the subsection II-D, learning a policy while selecting samples simultaneously with existing reinforcement learning framework could not be applied here. This framework requires many rounds of iterative selections, hence is against the time efficiency requirement for batch-mode active learning.

We learn the policy on a labelled dataset (called source dataset) in a simulation step, and then transfer the policy to select unlabeled samples on our target dataset in real-time environment. We start with a random policy and improve the policy network iteratively on the source dataset with imitation learning. Imitation learning is a machine learning technique that proves to be more useful when the expert can provide the optimal action easily as opposed to specifying a reward function that would generate the same action [24]. Imitation learning is a good fit for the simulation step in our framework, as the ground truth of which sample maximizes the base model

performance can be effortlessly found. In addition, imitation learning has been applied in single-point active learning [23] where the policy is trained to predict a probability vector to select the next single sample. Still, the application of imitation learning in batch-mode active learning has not been tackled.

Multiple episodes are simulated during the simulation phase. Within each episode, we randomly split the source dataset $C = \{x, y\}$ into an unlabeled sample pool C^{unl} and a validation sample pool C^{val}. We also create an empty labeled sample pool C^{lab} for a cold-start scenario. Alternatively, we can start from a non-empty C^{lab} as warm-start if we have labels. Finally, we need to create an empty dataset to record the pairs of current states and their corresponding ideal samples (denoted by SI) to train the policy.

For each simulation, we select Q samples per round for R rounds. At the beginning of each round, we start with an empty batch (selected sample pool) C^{sel}. Within this round, we keep selecting an ideal sample from C^{unl} and adding it to C^{sel} until we get Q samples in C^{sel}. Then we dump C^{sel} to C^{lab} and reset C^{sel} to empty. In particular, at round r to select the q-th sample, we define the ideal sample $x^*_{r,q}$ (size $n\_feature \times 1$) and its corresponding label $y^*_{r,q}$ as the sample that maximizes the base model m performance on the validation pool C^{val} when it combines with the current batch C^{sel} and already labeled samples C^{lab} (notice that we know the labels of C):

$$x^*_{r,q}, y^*_{r,q} = \\ argmax_{\{x,y\} \in C^{unl}} perf(m(C^{lab} \cup C^{sel} \cup \{x, y\}), C^{val}) \quad (3)$$

where $m(C^{lab} \cup C^{sel} \cup \{x, y\})$ is the base model trained by the dataset $C^{lab} \cup C^{sel} \cup \{x, y\}$, and $perf(m, C^{val})$ is m's performance on C^{val}, accuracy, F1-score, to name a few. We then add this state-ideal sample pair to SI.

However, since the states $s_{r,q}(C^{lab}, C^{sel}, C^{unl}, m)$ are not i.i.d, we cannot only use this ideal sample as the action we take to transfer to the next state. In the active learning process, the samples we should select actually depend on the samples we already selected. Thus we apply the DAGGER algorithm, which aims to solve this problem in imitation learning that future observations depend on previous predictions (actions) and violate the common i.i.d. assumptions made in statistical learning [25]. The DAGGER algorithm proves that when the number of simulations is large enough, the policy we get will be asymptotic to the optimal policy. It suggests that besides selecting the actual ideal sample x^* above, with some probability, we also select the predicted ideal sample $P_{r,q}$:

$$P_{r,q} = \pi(s_{r,q}) = \pi(s_{r,q}(C^{lab}, C^{sel}, C^{unl}, m)) \quad (4)$$

In practice, $P_{r,q}$ might not exist in the candidates pool C^{unl}, then the predicted action would be to select the unlabeled sample $\bar{x}$ that is closest to the predicted ideal sample:

$$\bar{x}_{r,q}, \bar{y}_{r,q} = argmin_{\{x,y\} \in C^{unl}} dist(x, P_{r,q}) \quad (5)$$

We can expedite this search process with approximate nearest neighbours methods [26], which is omitted in current research, yet worth future exploration. As we expect to apply this algorithm to select samples from a large size of unlabelled dataset, we assume that we can always find a pair of $\bar{x}_{r,q}, \bar{y}_{r,q}$ such that $dist(P_{r,q}, \bar{x}_{r,q}) < \delta$ for a negligibly small $\delta \geq 0$. In the experiments using publicly available data, we observe that the assumption holds quite well, and the policy converges eventually.

We define the loss l of policy π at the current state $s_{r,q}$ as

$$l(s_{r,q}, \pi) = dist(\pi(s_{r,q}), x^*_{r,q}) = dist(P_{r,q}, x^*_{r,q}) \quad (6)$$

The action we take in round r and step q (the sample-label pair to be added to C^{sel}) is:

$$A_{r,q} = I(coin_{t,r} = 0) * \{x^*_{r,q}, y^*_{r,q}\} + I(coin_{t,r} = 1) * \{\bar{x}_{r,q}, \bar{y}_{r,q}\} \quad (7)$$

where $I(A)$ is an indicator random variable that has value 1 if event A occurs and has value 0 otherwise; $coin$ is a random variable satisfying Bernoulli distribution:

$$coin_{t,r} = \begin{cases} 0 & : prob = w_{t,r} \\ 1 & : prob = 1 - w_{t,r} \end{cases} \quad (8)$$

To fulfill the prerequsites of DAGGER algorithm [25], $\{w_{t,r}\}$ is a converging sequence such that $\frac{1}{TR}\sum_{t,r} w_{t,r} \to 0$ as $T \to \infty$; T is the number of simulations and R denotes the number of rounds. For example, $w_{t,r}$ could be $p^t (0 < p < 1)$.

Figure 3 and Algorithm 1 summarize the simulation process for training an active learning policy. The learned policy is then transferred to select unlabelled samples for labelling on the target dataset according to Algorithm 2, and eventually output a trained base model; Table II puts together the definitions of symbols throughout the paper for reference.

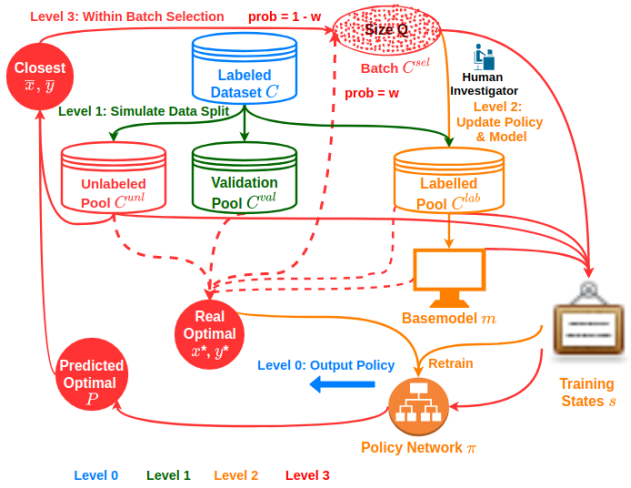


Fig. 3: Simulation step of NimbleLearn. In level 3 we update batch; in level 2 we update labeled pool and policy network; in level 1 we generate different data splits; in level 0 we output the policy network.

C. Convergence Guarantee and Algorithm Complexity

Training Convergence Guarantee: During training, the policy π is used to select $\bar{x}, \bar{y}$ each round, which can be different

Algorithm 1 Train Policy via Imitation Learning.

Input: source labeled data C, simulation episodes T, rounds R, query size Q, random variable sequence $coin_{t,r}$
Output: the learned policy π

Initialize a random base model m
SI = $\{\}$
for t = 1:T **do**
  Split C into C^{unl}, C^{val} randomly, $C^{lab} = \{\}$
  for r = 1:R **do**
    $C^{sel} = \{\}$
    for q = 1:Q **do**
      Get state $s(C^{lab}, C^{sel}, C^{unl}, m)$
      Predict ideal sample $P = \pi(s)$
      Closest $\bar{x}, \bar{y} = argmin_{\{x,y\} \in C^{unl}} dist(x, P)$
      Real ideal sample $x^*, y^* = argmax_{\{x,y\} \in C^{unl}} perf(m(C^{lab} \cup C^{sel} \cup \{x, y\}), C^{val})$
      Choose $\{x^*, y^*\}$,$\{\bar{x}, \bar{y}\}$, as selected $\{x^{sel}, y^{sel}\}$ with $coin_{t,r}$
      $C^{sel}.add(\{x^{sel}, y^{sel}\})$
      SI.add({state, x^*})
      C^{unl}.delete($\{x^{sel}, y^{sel}\}$)
    end for
    C^{lab}.add(C^{sel})
    m.retrain(C^{lab})
    π.retrain(SI)
  end for
end for
return policy π

Algorithm 2 Transfer Policy in Active Learning.

Input: unlabeled target dataset D^{unl}, round R, query size Q, policy π
Output: the learned base model m

Initialize a random base model m, empty labeled pool $D^{lab} = \{\}$
for r = 1:R **do**
  Selected pool D^{sel} = $\{\}$
  for q = 1:Q **do**
    Get state $s(D^{lab}, D^{sel}, D^{unl}, m)$
    Predict ideal sample $P = \pi(s)$
    Closest $\bar{x} = argmin_{x \in D^{unl}} dist(x, P)$
    $D^{sel}.add(\bar{x})$
    $D^{unl}.delete(\bar{x})$
  end for
  Get label for the samples in D^{sel} from investigators
  D^{lab}.add(D^{sel})
  $m.retrain(D^{lab})$
end for
return base model m

TABLE II: Definitions of symbols.

Symbol	Description
C, D	Source dataset, target dataset.
$C^{lab}, C^{sel}, C^{unl}, C^{val}$	Labeled/selected(batch)/unlabeled/validation source sample pool.
$D^{lab}, D^{sel}, D^{unl}$	Labeled/selected/unlabeled target sample pool.
$n\_feature$	Number of features in a sample.
$n\_percentile$	Number of percentiles in training states.
x_i, y_i	A $n\_feature \times 1$ sample and its label.
$P_{r,q}$	The $n\_feature \times 1$ ideal sample predicted by policy in round r, q.
$T(t), R(r), Q(q)$	# of simulations (t-th simulation)/rounds of selection (r-th round)/samples in a batch (q-th sample).
$A_{r,q}$	The sample selected in round r, q.
$x^*_{r,q}, y^*_{r,q}$	The real ideal sample that could improve base model performance at most at round r, q.
$\bar{x}_{r,q}, \bar{y}_{r,q}$	The sample in D^{lab} that is closest to the predicted ideal sample $P_{r,q}$ in round r, q.
m, π, s	base model (classifier), policy (active learning agent), training state (policy's input).

from the ideal sample $x^*_{r,q}, y^*_{r,q}$ selected by human expert. The difference can be quantified by:

$$\begin{aligned} loss(s_{r,q}, \pi) &= dist(\bar{x}_{r,q}, x^*_{r,q}) \\ &\leq dist(P_{r,q}, \bar{x}_{r,q}) + dist(P_{r,q}, x^*_{r,q}) \\ &< \delta + l(s_{r,q}, \pi) \end{aligned} \tag{9}$$

Let d_π be the average distribution of states if we follow policy π for Q steps (i.e, select Q samples in a batch), and $\epsilon = \mathbb{E}_{s \sim d_\pi}[loss(s, \pi)]$ be the expected surrogate loss during the process. Then by equation 9, $\epsilon < \delta + \mathbb{E}_{s \sim d_\pi}[l(s, \pi)]$. And according to the Theorem 4.1 in [25], $\mathbb{E}_{s \sim d_\pi}[l(s, \pi)]$ asymptotically converges to $\frac{1}{TR} \sum_{t,r} \mathbb{E}_{s \sim d_{\pi_{t,r}}}[l(s, \pi^*) + l(s, \pi_{t,r})] + O(\frac{1}{TR})$, where π^* is the ideal policy in hindsight after TR iterations, and $\pi_{t,r}$ denotes the learned policy after t-th simulation and r-th round. This completes the convergence of the learned policy π to ideal policy π^* after sufficient simulations.

Training and Transferring Complexity: Due to the carefully designed policy network, it avoids scaling with the enormous size of unlabelled samples. The whole training process consists of retraining policy network and base models for TR times, and finding optimal actions for TRQ times. Likewise, during the transferring process, it includes R times of retraining base models and RQ times of calculating distance and predicting actions. This is a significant reduction in runtime when compared with single-point active learning and existing batch-mode active learning approaches which involve $\binom{|D^{unl}|}{Q}$-size integer LP problems.

IV. Experiment

A. Experiment Setup

Tasks & Datasets. We evaluate NimbleLearn on 2 different tasks (spam email recognition and hate speech classification) with 4 public datasets outlined in Table III. For the spam email recognition, NimbleLearn is trained on an imbalanced dataset and tested on a balanced dataset. For the hate speech classification, it is trained on a balanced dataset and tested on an imbalanced dataset.

TABLE III: Summary of benchmark datasets and budgets.

Task	Source Dataset	Target Dataset	Total Budget
Spam Email Recognition	Enron Dataset [27]: 1500 spam emails and 3672 non-spam emails.	Nigerian Dataset [28]: 5187 spam emails and 6742 non-spam emails.	Around 5% of the target dataset, i.e., select 10 rounds and each round select 50 samples.
Hate Speech Classification	English Hateeval Dataset [29]: 4210 hateful tweets and 5790 non-hateful tweets.	Stormfront Dataset [30]: 932 hateful posts and 8527 non-hateful posts. We select those which the annotators does not ask for additional context.	Around 10% of the target dataset, i.e., select 10 rounds and each round select 100 samples.

Training & Testing Process. The specific training and testing processes are detailed as follow:

1) An active learning policy π is trained on a source dataset C (simulation step) and transferred onto a target dataset D for testing (transfer step).
2) In each episode of the simulation step during the training phase, 20% of the source dataset C is set as validation sample pool C^{val}, and the remaining 80% as the unlabeled sample pool C^{unl}. We choose bi-directional LSTM classifier [31] as the base model, and optimize the model against Macro-F1 score. The individual word vector input to the base model is embedded through a 300-dimension pre-trained Glove word vectors [32]. To quantify the cosine similarity (equation 5) between the predicted ideal sample and the candidate sample, we embed textual input with Sentence-BERT [33].
3) We run $T = 20$ episodes per simulation, and end up with collecting $20 \times 10 \times 50 = 10000$ $\{states, ideal\ \ sample\ \ P\}$ pairs of data points in total to train the policy network.
4) In the transfer step, 20% of target dataset D is set as test sample pool D^{test}, and 80% as unlabeled sample pool D^{unl}. After selecting and labelling each batch of samples, we retrain the base model m with all the labeled samples D^{lab} and evaluate the macro-F1 score of base model m on D^{test}. We repeat 10 times and report the average performance of strategies and the corresponding standard deviations in Table IV,V, VI.

Baselines. We compare NimbleLearn with other batch-mode active learning algorithms, ranging from heuristic methods (Random/Uncertainty/Diversity strategy) to methods that combine representativeness and informativeness (SPAL/BMDR/K-center). Their key concepts are summarized in the Appendix A.

B. NimbleLearn Performance in Different Scenarios

The trained base model and learned policy can have multiple options to be applied during the transfer step. Under three different scenarios, we evaluate the Macro-F1 score of the base model per round using learned NimbleLearn policy during the transfer step. The details of scenarios are summarized below.

- Direct transfer: we train a base model m on source dataset C and directly apply m to make prediction on target dataset D.
- Cold transfer: we train an active learning policy π on source dataset C and apply π to learn a new base model m on the target dataset D from scratch.
- Warm transfer: we train a base model m and an active learning policy π on source dataset C, and we continue improving m on the target dataset D by adding newly selected samples from π.

TABLE IV: Average F1-scores and standard deviations of base model on the test dataset D^{test} under different scenarios of NimbleLearn, with respect to the number of samples labeled.

Spam Email Recognition				Hate Speech Classification			
$\lvert D^{lab}\rvert$	Direct	Cold	Warm	$\lvert D^{lab}\rvert$	Direct	Cold	Warm
0	0.249±0.008	0.323±0.006	0.249±0.008	0	0.444±0.010	0.260±0.006	0.444±0.010
50		0.905±0.014	0.803±0.051	100		0.492±0.025	0.505±0.008
100		0.918±0.031	0.877±0.027	200		0.529±0.015	0.527±0.014
150		0.933±0.010	0.895±0.016	300		0.525±0.027	0.519±0.011
200		0.933±0.022	0.907±0.016	400		0.536±0.018	0.529±0.010
250		0.934±0.032	0.917±0.018	500		0.534±0.011	0.527±0.015
300		0.938±0.027	0.928±0.011	600		0.532±0.012	0.535±0.015
350		0.944±0.023	0.929±0.018	700		0.533±0.011	0.536±0.012
400		0.946±0.020	0.941±0.013	800		0.536±0.016	0.531±0.018
450		0.950±0.015	0.945±0.012	900		0.539±0.012	0.533±0.013
500		0.958±0.010	0.954±0.008	1000		0.530±0.010	0.530±0.012

Table IV summarizes the performance of NimbleLearn under different scenarios as we gradually label one batch of selected samples per round till the budget is met. Both cold-NimbleLearn and warm-NimbleLearn outperform direct transfer. In spam email recognition task, directly transferring the base model m on target dataset D only reaches 0.249 F1-score, whereas cold-NimbleLearn and warm-NimbleLearn reach over 0.9. Between the two NimbleLearn models, cold-NimbleLearn reaches 0.9 with 50 samples, while warm-NimbleLearn needs 200 samples to achieve comparable performance. In hate speech classification task, cold transfer base model has a lower starting score of 0.260, and gradually outperforms warm transfer base model with a score of 0.530. In both tasks, the cold-transfer base model achieves better performance with fewer samples, indicating transferring the knowledge of selecting samples is better than transferring trained base models, especially when a significant difference exists between the source dataset and target dataset.

C. NimbleLearn Performance VS Batch-mode Active Learning Baselines

Table V and Table VI summarize the Macro-F1 scores of the base model by applying NimbleLearn and various baseline batch-mode active learning strategies under the cold-start scenario for the spam email recognition and hate speech classification tasks respectively. In spam email recognition, NimbleLearn reaches 0.905 F1-score with 50 samples and 0.958 with 500 samples. Uncertainty and Diversity strategies reach over 0.97 with 500 samples, but need over 250 samples to reach 0.9. SPAL and K-Center reach 0.85 with 50 samples but could not improve further with more added samples. Random strategy initially has a good F1-score using 50 samples but fails to keep pace with the NimbleLearn trend. BMDR gets a score lower than NimbleLearn when the number of selected samples is less than 200. In hate speech classification, NimbleLearn

TABLE V: The average F1-scores and standard deviations of active learning strategies on the spam email recognition task versus the number of labeled samples. BMDR sometimes fail to solve its QP problem and could not generate a solution.

$\|D^{lab}\|$	NimbleLearn	Uncertainty	Diversity	Random	BMDR	SPAL	K-Center
0	0.323±0.006	0.323±0.006	0.323±0.006	0.323±0.006	0.323	0.323±0.006	0.323±0.006
50	0.905±0.014	0.359±0.004	0.359±0.004	0.889±0.009	0.887	0.881±0.027	0.853±0.025
100	0.918±0.031	0.361±0.005	0.442±0.086	0.906±0.008	0.905	0.874±0.048	0.869±0.017
150	0.933±0.010	0.385±0.025	0.685±0.222	0.925±0.007	0.914	0.876±0.041	0.858±0.022
200	0.933±0.022	0.590±0.172	0.841±0.160	0.934±0.006	0.942	0.871±0.037	0.868±0.020
250	0.934±0.032	0.848±0.056	0.910±0.082	0.936±0.007	0.943	0.878±0.042	0.850±0.038
300	0.938±0.027	0.927±0.018	0.953±0.012	0.953±0.007	0.949	0.873±0.042	0.862±0.019
350	0.944±0.023	0.946±0.012	0.960±0.008	0.954±0.006	0.932	0.869±0.035	0.850±0.029
400	0.946±0.020	0.957±0.012	0.964±0.006	0.952±0.007	0.960	0.867±0.038	0.854±0.029
450	0.950±0.015	0.966±0.005	0.970±0.005	0.953±0.005	0.960	0.870±0.041	0.857±0.027
500	0.958±0.010	0.972±0.006	0.973±0.003	0.948±0.004	0.958	0.873±0.044	0.857±0.026

TABLE VI: The average F1-scores and standard deviations of active learning strategies on the hate speech classification task given the number of samples labeled. BMDR sometimes fail to solve its QP problem and could not generate a solution.

$\|D^{lab}\|$	NimbleLearn	Uncertainty	Diversity	Random	BMDR	SPAL	K-Center
0	0.260±0.006	0.260±0.006	0.260±0.006	0.260±0.006	0.260	0.260±0.006	0.260±0.006
100	0.492±0.025	0.475±0.002	0.475±0.002	0.480±0.014	0.475	0.485±0.025	0.475±0.002
200	0.529±0.015	0.477±0.005	0.483±0.015	0.513±0.027	0.511	0.510±0.017	0.491±0.028
300	0.525±0.027	0.496±0.024	0.495±0.017	0.526±0.025	0.513	0.507±0.017	0.494±0.030
400	0.536±0.018	0.499±0.022	0.486±0.010	0.513±0.017	0.515	0.513±0.021	0.487±0.027
500	0.534±0.011	0.498±0.015	0.490±0.010	0.518±0.021	0.510	0.509±0.021	0.490±0.026
600	0.532±0.012	0.511±0.016	0.489±0.013	0.507±0.023	0.511	0.517±0.019	0.486±0.023
700	0.533±0.011	0.498±0.014	0.488±0.011	0.510±0.019	0.513	0.516±0.023	0.489±0.030
800	0.536±0.016	0.502±0.012	0.493±0.014	0.527±0.025	0.509	0.505±0.018	0.487±0.026
900	0.539±0.012	0.507±0.009	0.488±0.012	0.516±0.013	0.516	0.505±0.021	0.483±0.019
1000	0.530±0.010	0.505±0.016	0.489±0.006	0.517±0.014	0.522	0.516±0.021	0.487±0.022

only requires 200 carefully selected samples to reach 0.53 (notably, the best macro-F1 score in the previous literature on this highly imbalanced Stormfront dataset [34] is 0.49 with a bi-LSTM model using all samples). All the other batch-model active learning baselines fluctuate around 0.5. This encouraging result demonstrates that NimbleLearn can adapt to imbalanced datasets, and assign proper weight in selecting representing samples.

D. Runtime Analysis

In this subsection, we compare the runtime during transfer step by applying NimbleLearn versus single-point active learning with uncertainty strategy and aforementioned batch-model active learning baselines.

Assuming all active learning agents take unit and negligible time selecting samples. This assumption holds true as long as the main contributors to the runtime in practice are refreshing the base model per round and manually labelling samples. We assume model refresh takes t^{trn}, and it increases linearly with input size, i.e. $t^{trn} = c_{trn} * |D^{lab}|$, where c_{trn} is a positive coefficient. The conclusion still holds if t^{trn} increases faster than linear as single-point active learning needs more rounds of selection. In the single-point active learning, the base model is refreshed upon every newly added sample, whereas the base model is refreshed upon every newly added batch under the batch-model active learning setting. Manual labelling process takes t^{lab} per round and c_{lab} per sample respectively. This equation holds $t^{lab} = |D^{sel}|/n\_inv * c_{lab}$ between t^{lab} and c_{lab}, where n_inv denotes manual capacity. The total runtime t^{tot} of the active learning process is found to satisfy (see Appendix B):

$$t^{tot} = (\frac{R+1}{2}c_{trn} + \frac{c_{lab}}{n\_inv})|D^{lab}| \tag{10}$$

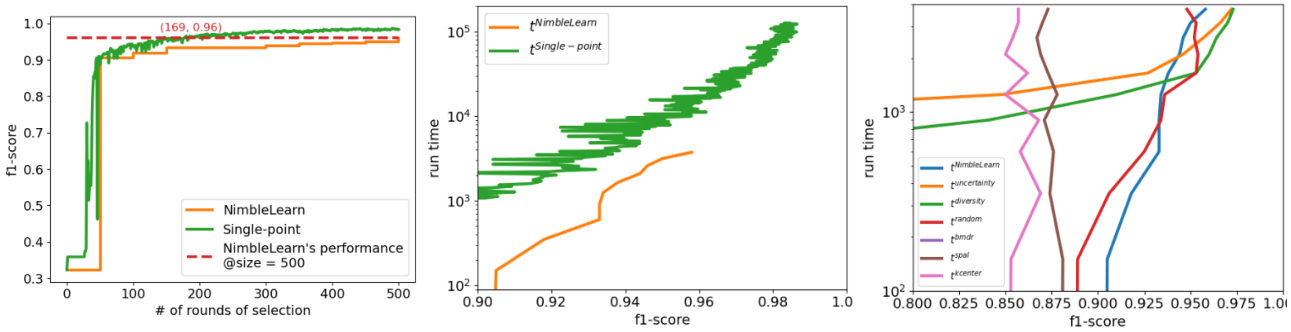
Figure 4a plots the F1-score of single-point active learning with uncertainty strategy applied on spam email recognition task. It needs 169 rounds of selection to have the base model achieve performance of 0.96, versus 10 rounds for NimbleLearn. It is worth noting that single-point active learning achieves the result using 169 labelled samples, while 500 labelled samples for NimbleLearn. This difference in sample size can be explained by the fact that single-point active learning selects the most optimal sample each round, and often will yield better performance using fewer labelled samples in total. Plugging in the specific number into the t^{tot} formula, one can easily prove that NimbleLearn always outperforms the single-point active learning in terms of runtime when 3 or more human investigators are available (see Appendix B). Figure 4b depicts the runtime between NimbleLearn and single-point active learning when $n\_inv = 50$, $c_{trn} = 1$ and $c_{lab} = 100$ for illustration purpose. NimbleLearn indeed requires considerably less time than the single-point active learning to achieve a same F1-score level.

Interestingly, in hate speech classification task, the single-point active learning can never reach performance of 0.53 as NimbleLearn does, see Figure 5a, 5b. In the spam email recognition task, single-point active learning selects the optimal sample each round, in many if not all cases it should yield better performance than NimbleLearn. However, such greedy process gets stuck in local optimization as the heuristic fails for imbalanced target dataset. This finding indicates NimbleLearn can better avoid local optimization for highly imbalanced target dataset.

In addition, we compare the runtime of NimbleLearn with other batch-mode active learning strategies. We set $n\_inv = 50$, $c_{trn} = 1$, $c_{lab} = 100$ for illustration purpose, and apply equation 10. For the spam email recognition task, to reach a reasonable F1-score of 0.9, NimbleLearn requires less runtime in some ranges of F1-score compared to other algorithms, see Figure 4c and further discussion below. For the hate speech classification task, other batch-mode active learning algorithms can not reach the same model performance as NimbleLearn does, and NimbleLearn also achieves lowest runtime consistently (see Figure 5c).

E. Robustness Analysis

Unlike many other batch-mode learning methods that require pre-defined parameters such as weights, "informativeness" or "diversity" scores, NimbleLearn only requires two parameters: number of rounds R and batch size Q. The exact values of R and Q assigned to the experiment have a small influence on our conclusion, as shown by the robustness analysis conducted on in spam email detection (Figure 6a). In spam email detection, the ultimate performance of NimbleLearn is relatively stable regarding the batch size Q and rounds of

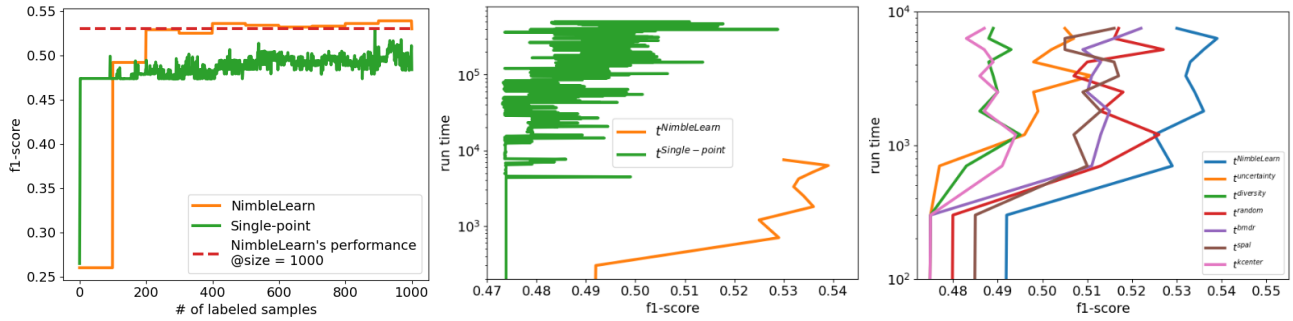


(a) Performance of single-point active learning on spam email recognition task.

(b) Single-point strategy and NimbleLearn's runtime against F1-score on spam email task.

(c) Batch-mode Algorithms' runtime against F1-score on the spam email recognition task.

Fig. 4: Performance and runtime of NimbleLearn and active learning algorithms on spam email recognition task.



(a) Performance of single-point active learning on hate speech classification task.

(b) Single-point strategy and NimbleLearn's runtime against F1-score on hate speech task.

(c) Batch-mode Algorithms' runtime against F1-score on hate speech classification task.

Fig. 5: Performance and runtime of NimbleLearn and active learning algorithms on hate speech classification task.

selections R, and NimbleLearn still has good performance even when we try to learn the pattern of large batches. The hate speech detection task is more sensitive to batch size (Figure 6b). For instance, when $Q = 20, 50$, the performance can not be stably improved along with the number of labeled samples, which might be caused by the great imbalance of the target dataset.

Similarly, we quantify NimbleLearn sensitivity regarding the target dataset imbalance degree (the ratio of the number of the negative samples to the number of positive samples). In Figure 6c, we plot the final F1-score after using NimbleLearn to select 100 samples per round for 10 rounds, against the dataset imbalance degree. Overall, NimbleLearn is not sensitive to the degree of dataset imbalance.

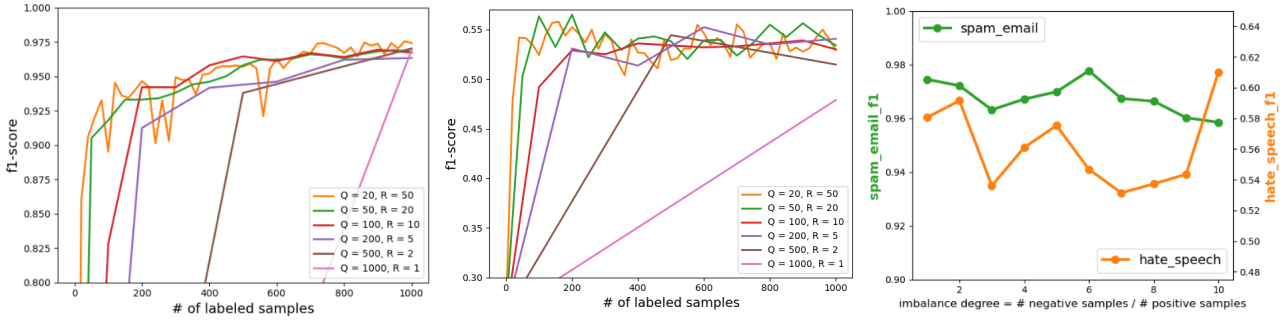
V. Discussion

In the transferring process, NimbleLearn in general requires fewer manually labeled samples to reach a reasonably good performance than other batch-mode active learning algorithms, resulting in a runtime reduction. When manual labelling budget and runtime are tightly constrained, NimbleLearn is the best strategy among all the tested batch-mode alternatives; For instance, when the allocated runtime threshold is 2000 (Figure 4c). However, when runtime threshold is relaxed to larger than 2000 (i.e. label more samples), diversity strategy can yield better F1-score.

NimbleLearn performs far better than other batch-mode active learning algorithms on highly imbalanced datasets. Unsurprisingly, NimbleLearn is designed to learn an agent that can improve the desired metric of the base model based on the target metric directly, whereas alternative batch-mode active learning algorithms select samples based on heuristics or assumptions, which can not optimize the target metric directly for imbalanced dataset. In spam email recognition, where the target dataset is balanced, batch-mode active learning algorithms can achieve similar performance as NimbleLearn does. However, NimbleLearn easily outperforms baselines in the imbalanced hate speech detection task.

VI. Conclusion

In this paper, we propose NimbleLearn, a parameter-free batch-mode active learning algorithm, to select a subset of unlabelled samples efficiently when the sample size is huge. It resolves the scalability and speed issues in existing SOTA approaches. NimbleLearn trains a policy network to predict the "ideal sample" greedily, and this ideal sample is added to the selected batch to maximize the performance of any base model in use. Through experiments conducted on two different



(a) F1-score of NimbleLearn on spam email recognition task given different rounds of selection (R) and batch size (Q).

(b) F1-score of NimbleLearn on hate speech classification task given different rounds of selection (R) and batch size (Q).

(c) Final F1-score of NimbleLearn on both tasks given different imbalance degrees of the target datasets.

Fig. 6: Robust test of NimbleLearn on both tasks.

tasks, NimbleLearn can yield a better base model performance with fewer rounds of selection, and result in an enhancement in overall runtime.

REFERENCES

[1] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[2] ——, "From theories to queries: Active learning in practice," in *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, 2011, pp. 1–18.

[3] S. C. Hoi, R. Jin, and M. R. Lyu, "Batch mode active learning with applications to text categorization and image retrieval," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1233–1248, 2009.

[4] ——, "Large-scale text categorization by batch mode active learning," in *Proceedings of the 15th international conference on World Wide Web*, 2006, pp. 633–642.

[5] Y.-P. Tang and S.-J. Huang, "Self-paced active learning: Query the right thing at the right time," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5117–5124.

[6] Z. Wang and J. Ye, "Querying discriminative and representative samples for batch mode active learning," vol. 9, no. 3, Feb. 2015. [Online]. Available: https://doi.org/10.1145/2700408

[7] T. N. Cardoso, R. M. Silva, S. Canuto, M. M. Moro, and M. A. Gonçalves, "Ranked batch-mode active learning," *Information Sciences*, vol. 379, pp. 313 – 337, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025516313949

[8] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *SIGIR'94*. Springer, 1994, pp. 3–12.

[9] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, "Selective sampling using the query by committee algorithm," *Machine learning*, vol. 28, no. 2, pp. 133–168, 1997.

[10] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of machine learning research*, vol. 2, no. Nov, pp. 45–66, 2001.

[11] A. McCallum, K. Nigam *et al.*, "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752, no. 1. Citeseer, 1998, pp. 41–48.

[12] I. Muslea, S. Minton, and C. A. Knoblock, "Active+ semi-supervised learning= robust multi-view learning," in *ICML*, vol. 2. Citeseer, 2002, pp. 435–442.

[13] K. Brinker, "Incorporating diversity in active learning with support vector machines," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 59–66.

[14] Z. Xu, K. Yu, V. Tresp, X. Xu, and J. Wang, "Representative sampling for text classification using support vector machines," in *European conference on information retrieval*. Springer, 2003, pp. 393–407.

[15] Y. Li, Y. Wang, D.-J. Yu, N. Ye, P. Hu, and R. Zhao, "Ascent: Active supervision for semi-supervised learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 5, pp. 868–882, 2019.

[16] S. C. Hoi, R. Jin, J. Zhu, and M. R. Lyu, "Batch mode active learning and its application to medical image classification," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 417–424.

[17] S.-J. Huang, R. Jin, and Z.-H. Zhou, "Active learning by querying informative and representative examples," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 10, pp. 1936–1949, 2014.

[18] S. Chakraborty, V. Balasubramanian, Q. Sun, S. Panchanathan, and J. Ye, "Active batch selection via convex relaxations with guaranteed solution bounds," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 10, pp. 1945–1958, 2015.

[19] R. Chattopadhyay, Z. Wang, W. Fan, I. Davidson, S. Panchanathan, and J. Ye, "Batch mode active sampling based on marginal probability distribution matching," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 7, no. 3, pp. 1–25, 2013.

[20] W. Fu, M. Wang, S. Hao, and X. Wu, "Scalable active learning by approximated error reduction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, ser. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1396–1405. [Online]. Available: https://doi.org/10.1145/3219819.3219954

[21] Z. Liu, J. Wang, S. Gong, H. Lu, and D. Tao, "Deep reinforcement active learning for human-in-the-loop person re-identification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6122–6131.

[22] M. Lopes, F. Melo, and L. Montesano, "Active learning for reward estimation in inverse reinforcement learning," in *Machine Learning and Knowledge Discovery in Databases*, W. Buntine, M. Grobelnik, D. Mladenić, and J. Shawe-Taylor, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 31–46.

[23] M. Liu, W. Buntine, and G. Haffari, "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)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 1874–1883. [Online]. Available: https://www.aclweb.org/anthology/P18-1174

[24] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[25] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[26] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration." in *VISAPP (1)*, 2009, pp. 331–340.

[27] B. Klimt and Y. Yang, "Introducing the enron corpus." in *CEAS*, 2004. [Online]. Available: https://www.cs.cmu.edu/ enron/

[28] D. Radev, "Clair collection of fraud email, acl data and code repository," *ADCR2008T001*, 2008. [Online]. Available: https://www.kaggle.com/llabhishekll/fraud-email-dataset

[29] V. Basile, C. Bosco, E. Fersini, D. Nozza, V. Patti, F. M. Rangel Pardo, P. Rosso, and M. Sanguinetti, "SemEval-2019 task 5:

Multilingual detection of hate speech against immigrants and women in Twitter," in *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, Jun. 2019, pp. 54–63. [Online]. Available: https://github.com/msang/hateval

[30] O. de Gibert, N. Perez, A. García-Pablos, and M. Cuadros, "Hate Speech Dataset from a White Supremacy Forum," in *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 11–20. [Online]. Available: https://github.com/Vicomtech/hate-speech-dataset

[31] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[32] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[33] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019.

[34] H. S. Alatawi, A. M. Alhothali, and K. M. Moria, "Detecting white supremacist hate speech using domain specific word embedding with deep learning and bert," *arXiv preprint arXiv:2010.00357*, 2020.

[35] Y.-P. Tang, G.-X. Li, and S.-J. Huang, "ALiPy: Active learning in python," Nanjing University of Aeronautics and Astronautics, Tech. Rep., Jan. 2019, available as arXiv preprint https://arxiv.org/abs/1901.03802. [Online]. Available: https://github.com/NUAA-AL/ALiPy

[36] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," *arXiv preprint arXiv:1708.00489*, 2017.

APPENDIX A
BATCH-MODE ACTIVE LEARNING BASELINES

- Random strategy: selecting Q samples from the unlabeled pool D^{unl} randomly with equal probability in each round.
- Uncertainty strategy: selecting top Q samples where the base model has the "least confident score" in each round. The "least confident score" of sample x is defined as:

$$least\_confident\_score(x) = 1 - max(m.predict(x)) \quad (11)$$

  where the prediction results of m is a probability vector of each class.
- Diversity strategy [7]: selecting the sample with the highest "combination score", which is a weighted sum of least confident score and diversity score with preset parameters:

$$combination\_score(x) = \alpha * diversity\_score(x, D^{sel}) + (1-\alpha) * least\_confident\_score(x) \quad (12)$$

  We set $\alpha = 0.5$ for the experiment. The diversity score is measured by the candidate sample's distance to the batch of samples already selected in the current round D^{sel}:

$$diversity\_score(x, D^{sel}) = \min_{x' \in D^{sel}} (1 - cosine\_sim(x', x)) \quad (13)$$

- SPAL [5]: selecting a batch of informative, representative and easy examples by a self-paced approach. It tries to train the base model with the least cost by "querying the right thing at the right time".
- BMDR [6]: selecting a batch of informative and representative examples by minimizing the ERM risk bound of active learning. This method involves solving a quadratic programming problem multiple times at one query, and is time-consuming for large datasets [35].
- K-Center [36]: a batch-mode active learning strategy that selects the center samples to minimize the largest distance between each sample and its nearest center.

APPENDIX B
COST ANALYSIS

The total time spent in round r is

$$\begin{aligned} t_r &= t_r^{trn} + t_r^{lab} \\ &= c_{trn} * r * |D^{sel}| + c_{lab} * |D^{sel}|/n\_inv \end{aligned} \quad (14)$$

The total spent from round 1 to round r is:

$$\begin{aligned} t_r^{tot} &= \sum_{i=1}^{r} t_i = \sum_{i=1}^{r} (c_{trn} * i * |D^{sel}| + c_{lab} * |D^{sel}|/n\_inv) \\ &= (\frac{r+1}{2} c_{trn} + \frac{c_{lab}}{n\_inv}) r |D^{sel}| \end{aligned} \quad (15)$$

The total time of the active learning process ($r = R$) is:

$$t^{tot} = (\frac{R+1}{2} c_{trn} + \frac{c_{lab}}{n\_inv}) |D^{lab}| \quad (16)$$

We estimate how much time NimbleLearn spends compared to single-point active learning when they reach the same performance F1-score = 0.96 (see Figure 4a). NimbleLearn uses 10 rounds. The time NimbleLearn spent to reach 0.96 is:

$$t^{NimbleLearn} = (\frac{11}{2} c_{trn} + \frac{c_{lab}}{n\_inv}) * 500 \quad (17)$$

single-point active learning uses 169 rounds. The time single-point active learning spent to reach 0.96 is:

$$t^{single-point} = (\frac{170}{2} c_{trn} + c_{lab}) * 169 \quad (18)$$

The difference between $t^{single-point}$ and $t^{NimbleLearn}$ is:

$$\begin{aligned} t^{single-point} - t^{NimbleLearn} &= 11615 * c_{trn} - (\frac{500}{n\_inv} - 169) * c_{lab} \\ &= c_{lab}(11615 \frac{c_{trn}}{c_{lab}} - \frac{500}{n\_inv} + 169) \end{aligned} \quad (19)$$

First we could notice that when $n\_inv >= 3$, $t^{single-point} - t^{NimbleLearn}$ is always > 0 because $-\frac{500}{n\_inv} + 169 > 0$. It means that when we have 3 or more human investigators, to save runtime, we should always choose NimbleLearn.

When $n\_inv = 2$ (we only have 2 human investigators who could work in parallel), that means $-\frac{500}{n\_inv} + 169 = -81$, we should still select NimbleLearn when $\frac{c_{trn}}{c_{lab}} > \frac{81}{11615}$.

When $n\_inv = 1$ (we only have 1 human investigators and could not work in parallel), that means $-\frac{500}{n\_inv} + 169 = -331$, we should still select NimbleLearn when $\frac{c_{trn}}{c_{lab}} > \frac{331}{11615}$.