Towards Low-Budget Real-Time Active Learning for Text Classification via Proxy-Based Data Selection

Jakob Smedegaard Andersen and Olaf Zukunft
Department of Computer Science, Hamburg University of Applied Science, Germany

Keywords: Text Classification, Active Learning, Cost-Sensitive Learning.

Abstract: Training data is typically the bottleneck of supervised machine learning applications, heavily relying on cost-intensive human annotations. Active Learning proposes an interactive framework to efficiently spend human efforts in the training data generation process. However, re-training state-of-the-art text classifiers is highly computationally intensive, leading to long training cycles that cause annoying interruptions to humans in the loop. To enhance the applicability of Active Learning, we investigate low-budget real-time Active Learning via Proxy-based data selection in the domain of text classification. We aim to enable fast interactive cycles within a minimal labelling effort while exploiting the performance of state-of-the-art text classifiers. Our results show that Proxy-based Active Learning can increase the F1-score of a lightweight classifier compared to a traditional budget Active Learning approach up to $\sim$19%. Our novel Proxy-based Active Learning approach can be carried out time-efficiently, requiring less than 1 second for each learning iteration.

1 INTRODUCTION

The acquisition of training examples is an integral part of a classifier's learning process (Yang, 1999). While unlabelled data is nowadays commonly available, the general lack of labelled data-instances forms a bottleneck. Human annotators are needed to manually label and extend the training corpus until a sufficient amount of data is acquired (Fails and Olsen Jr, 2003). However, human involvement is typically time and cost-intensive. In order to save human efforts, it is desirable to only manually label instances which highly contribute to a model's learning behaviour.

Active Learning (AL) (Settles, 2009) provides a framework to efficiently spend human efforts in the training data generation process by reducing the number of annotations needed (Lewis and Gale, 1994). The overall idea of AL is simple: a small but informative set of training examples can lead to the same or even better performance than larger and noisier training data collections. In AL, a classifier iteratively queries additional labels from human annotators and is frequently re-trained when new training data is available. AL comes with a special time-related requirement to maintain its applicability in real-world settings (Settles, 2011). Fast interaction cycles are considerably more important for interactive Machine Learning (ML) approaches than their actual accuracy (Fails and Olsen Jr, 2003). In the domain of text classification, transformer-based models such as BERT (Devlin et al., 2019) and its variations provide state-of-the-art accuracies. However, BERT consists of hundreds of millions of parameters requiring very long training and inference durations. Enormous waiting times and interrupts emerge when using BERT within AL, negatively impacting the overall user experience (Doherty and Sorenson, 2015). In order to maintain fast interaction cycles, less complex and usually far less accurate classifiers have to be used instead, to reduce the latency of AL in real-world applications.

In this work, we study the potential of Proxy-based AL in the context of text classification. We aim to drastically reduce the computational duration time of AL while still taking advantage of state-of-the-art text classifiers during deployment. In a traditional AL process, the same classifier is used for data selection and deployment (Settles, 2009). Contrary, we investigate whether the data selection step can be carried out in real-time by a very fast but usually weak classification algorithm (called Proxy) to accelerate the overall AL process. Afterwards, the collected training data is used to train a state-of-the-art target classifier (called Consumer), i.e. BERT, which is then used in deployment. In this paper, we investigate the following research questions:

Andersen, J. and Zukunft, O.
Towards Low-Budget Real-Time Active Learning for Text Classification via Proxy-Based Data Selection.
DOI: 10.5220/0011606000003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 3*, pages 25-33
ISBN: 978-989-758-623-1; ISSN: 2184-433X

RQ1: How accurate are BERT classifiers trained via low-budget Proxy-based Active Learning?

RQ2: How big is the gain in accuracy of using Proxy-based Active Learning compared to traditional Active Learning using a budget classifier for data selection?

RQ3: How suitable is Proxy-based Active Learning for real-time processing?

The remainder of this paper is structured as follows. We shed light on related work in Section 2. Section 3 discusses our Proxy-based AL approach. Then, Section 4 introduces state-of-the-art candidate classifiers to implement Proxy-based AL. Section 5 outlines our evaluation design and Section 6 presents our results. We discuss our findings in Section 7. Finally, Section 8 outlines the conclusion.

2 RELATED WORK

The idea of using two distinct algorithms to train a classifier was initially discussed by Tomanek and Morik (Tomanek and Morik, 2011). They raise the so called re-usability problem of AL, which is about whether *"a set of labelled examples that is deemed most informative using one classification algorithm necessarily informative for another classification algorithm?"* They found that using traditional machine learning text classifiers, foreign selection is around 75% of the cases better than a random selection strategy. Hu et al. (Hu et al., 2016) investigate the mutual re-usability of pairs of traditional text classifiers. They investigate which combination of Proxy and Consumer provides the best performances. In contrast to our work, they do not consider the time savings of less complex Proxies. Lowell et al. (Lowell et al., 2019) investigate Proxy-based data sampling between similar accurate classifiers, including Deep Neural Networks (DNN). We focus on the transferability between a fast classifier and a state-of-the-art Deep Learning approach. Coleman et al. (Coleman et al., 2020) investigate the time-saving and error using a FastText classifier as the Proxy and a DNN as the Consumer. They show that FastText is up to 41.9 times faster while causing no significant error increase and does not harm the accuracy. However, their approach still requires multiple minutes for training, which is too slow for real-time processing. Prabhu et al. (Prabhu et al., 2019) also investigate Proxy-based AL using FastText, but they rely on very large labelling budgets. In contrast, we focus on a low-budget real-time setting where no more than 500 instances are queried during the AL process.

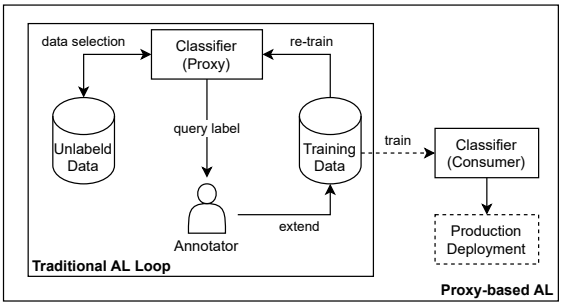

Figure 1: Proxy-based AL as an extension of the traditional AL loop.

3 PROXY-BASED AL FOR TEXT CLASSIFICATION

3.1 Text Classification

Text classification deals with the automatic categorization of text documents into predefined classes (Yang, 1999). The learning process of a classifier is supervised by a set of training examples $D_{train} \subset X \times Y$ consisting of text documents $x_1, ..., x_N \in X$ and associated class labels $y_1, ..., y_N \in Y$. A classifier can be seen as a function $f : X \rightarrow Y$ predicting a class label $y \in Y$ for new data samples $x \in X$ which are related according to an unknown class probability $p(y|x)$. Our study focuses on classification tasks where each document $x \in X$ belongs exclusively to one class. Typically, a large amount of manually labelled training examples have to be acquired to accurately train a classifier f. However, human time is typically sparse and can easily be too expensive to be used in real-world applications. Therefore, we strive to reduce human involvement within the training data generation process (Lewis and Gale, 1994).

3.2 Active Learning

Active Learning (AL) (Settles, 2009) is an interactive approach to reduce the number of annotations needed in order to adequately train classifiers and thus drastically hasten the learning process (Dor et al., 2020). The traditional AL process is depicted in the left part of Figure 1. In AL, a classifier selects instances which are likely to contribute the most to its learning behaviour. Labels of the selected data instances are queried from human annotators and are used to frequently re-train the model. In this work, we adopt *pool-based* AL (Sugiyama and Nakajima, 2009) where a large pool of unlabelled text documents is available. In this setting, AL aims to select a training data set which is likely to provide the best model

improvements when used as training data.

3.3 Proxy-Based AL

Traditional AL uses the same classification algorithm for data selection and deployment (self-selection). However, in many situations, self-selection is usually not applicable or desired (Tomanek and Morik, 2011). In real-world domains, rapid AL cycles are essential to continuously integrate humans in the AL loop (Fails and Olsen Jr, 2003). State-of-the-art classifiers like BERT are too complex for rapid AL, because they train and infer very slowly. Practitioners must choose less complex and thus usually less accurate models to meet time constraints.

In this paper we adapt Proxy-based AL (Tomanek and Morik, 2011; Coleman et al., 2020) which incorporates two different classifiers in its learning process. As illustrated by Figure 1, one classifier is dedicated to the selection of data (called *Proxy*). The secondary classifier is trained on the acquired dataset (*Proxy* sampling) and used in deployment (called *Consumer*). To hasten the data selection process, the Proxy has to be much faster, typically less complex and generally provides a less accurate result. In comparison, the Consumer should be a state-of-the-art classifier, likely to provide the best classification results on the gathered dataset.

4 TEXT CLASSIFIERS FOR PROXY-BASED REAL-TIME AL

4.1 Proxy and Consumer Candidates

A Proxy-candidate has to be computationally efficient while providing a reusable set of training data. In this work, we apply FastText (Joulin et al., 2017) and Logistic Regression as the Proxy. **FastText** is a simplistic neural network-based linear text classifier, which represents sentences by averaging trainable word vectors. **Logistic Regression** for text classification has shown to be very fast and more accurate than other budget classifiers such as Decision Trees, Naive Bayes and *k*-Nearest-Neighbours classifiers (Pranckevičius and Marcinkevičius, 2017). While FastText is based on a bag of words and bigrams, Logistic Regression models require meaningful feature vectors as inputs. In our experiments, we utilize state-of-the-art pre-trained language models (Qiu et al., 2020) to extract meaningful feature representations, i.e. embeddings, from text instances. We consider BERT (Devlin et al., 2019), Sentence-BERT (SBERT) (Reimers and Gurevych, 2019), and ELMo (Peters et al., 2018). Embeddings are pre-computed for all data-instances, cached and queried during the AL process.

We deploy **BERT** as the Consumer in our investigation of Proxy-based AL, as BERT has shown to reach state-of-the-art accuracies across a range of text classification tasks (Devlin et al., 2019). BERT is pre-trained on an unlabelled text corpus and can be directly incorporated in a downstream classification task by adding a softmax-function on top to predict the probabilities of class labels. BERT requires task specific fine-tuning using labelled data of the target domain to reach state-of-the-art accuracies (Devlin et al., 2019). Its default configuration consists of hundreds of million parameters, making it very resource hungry and slow in training and inference.

4.2 Data Selection Strategies

A selection strategy aims to identify the most informative text instances which are likely to have the highest impact on a model's accuracy when used for re-training. We focus on selection strategies which either perform very fast during the AL process or which can be calculated prior training to not affect the total duration of each AL iteration. We focus on the following selection strategies, including a baseline:

- **Random**: Selects instances uniformly at random.
- **Uncertainty**: Data instances are selected based on where the model is most uncertain about in regard to its labelling (Lewis and Catlett, 1994). We apply margin sampling (Scheffer et al., 2001) to quantify the uncertainty of predictions, which is $U(x) = \arg\min_x P(y_1|x) - P(y_2|x)$ where y_1 and y_2 are the first and second most probable class labels of a text input x.
- **Density*Uncertainty**: Since highly uncertain instances might not be representative, Zhu et al. (Zhu et al., 2008) suggest selecting instances according to the maximum uncertainty and the highest representatives in terms of density. The authors suggest a k-Nearest-Neighbour density measurement to evaluate the density of an instance $i \in D_{pool}$ in respect to the K most similar examples $S(x) = \{s_i\}_{i=1}^{K}$, which is defined as:

$$DS(x) = K^{-1} \sum_{s_i \in S(x)} \cos(x, s_i) \qquad (1)$$

  where $\cos(x, s_i)$ is the cosine-similarity between x and s_i. The *density*uncertainty* score function is defined as $DSH(x) = DS(x) \times U(x)$.
- **Instability**: Zhu et al. (Zhu and Ma, 2012) suggest an instability-based selection strategy.

The instability of a prediction is measured based on the changes of the predictive uncertainty scores during recent (n) consecutive learning cycles. The most unstable instances are those with the highest numerical changes. The authors provide two instability measurements, namely *Label-Insensitive Instability Sampling* (IS_{LI}) and *Label-Sensitive Instability Sampling* (IS_{LS}). IS_{LI} reports high uncertainties when instances cause unstable uncertainty estimates during the l most recent consecutive learning cycles, that is:

$$IS_{LI}(x) = U^i(x) + \sum_{i-l<k\leq i} U^k(x) - U^{k-1}(x) \quad (2)$$

IS_{LS} selects the most informative example from the set of unlabelled examples that have a high instability and different label predictions during recent consecutive learning cycles, that is:

$$IS_{LS}(x) = U^i(x) + \sum_{i-l<k\leq i} \delta(y^k, y^{k-1}) \times \left(U^k(x) - U^{k-1}(x)\right) \quad (3)$$

- **Density*Instability**: (Zhu and Ma, 2012) also suggest selecting instances according to the maximum instability, that is $DSIS_{LI}(x) = DS(x) \times IS_{LI}(x)$ and highest density which can be formulated as: $DSIS_{LS}(x) = DS(x) \times IS_{LS}(x)$.

5 EVALUATION DESIGN

5.1 Datasets Used for Evaluation

We consider three publicly available English datasets covering a variety of domains. The datasets are summarized in Table 1. First, we use the **App Store** dataset from the domain of participatory requirements engineering (Maalej et al., 2016). The dataset consists of manually labelled app reviews covering *feature requests*, *bug reports* and *praise*. Second, we utilize the **Hate Speech** (Davidson et al., 2017) dataset, which consists of tweets manually labelled for their toxicity (toxic / non-toxic). Third, we use the **Reuters** dataset (Lewis et al., 2004) consisting of highly unbalanced topic modelling tasks. In our experiments, we select a subset of the 9 most frequent topics with unambiguous labels. We split all the datasets into a 50% training and 50% test set, keeping the original label distribution. The training set is used as the pool for data selection. Training is then only performed on the selected instances. For the training of the BERT classifier, we use 10% of the selected training data as a validation set.

Table 1: Statistics of the datasets used.

Dataset	Size	$\|C\|$	Class Distribution	#Words ($\mu \pm \sigma$)
App Store	6392	3	3855:1437:1100	24 ± 29
Hate Speech	24783	2	20620:4163	14 ± 7
Reuters	8759	9	3930:2319:527:495: 458:425:282:166:157	152 ± 176

5.2 Evaluation Criteria

To answer our research questions, we first investigate the predictive performance of Proxy-based AL using FastText and Logistic Regression as the *Proxy* and BERT as the *Consumer*. Our investigation covers three datasets, two classifiers, three text embeddings and seven query strategies. Second, we investigate the performance improvements of a low-budget Proxy-based AL approach (Bert as the Consumer; FastText and Logistic Regression as Proxy) compared to traditional AL. We evaluate the performance gains the training of an additional state-of-the-art Consumer provides and whether it is worth the effort. Third, we investigate the real-time ability of our Proxy-based AL approach to assess its applicability in real-world settings. We measure the time needed to perform a learning iteration consisting of model training, inference and data selection. We follow the rule of thumb, that a user in an interactive setting should not wait more than 1 second for the response of his or her action, to maintain user experience and productivity (Tolia et al., 2006; Martin and Corl, 1986).

5.3 Evaluation Setup

For each experiment, we report the mean of 5 independent model runs based on stratified train-test splits. For the App Store and Hate Speech datasets, we randomly select 10 instances per class and 3 instances for the Reuters dataset. Leading to an initial training dataset of 30, 20 and 27 instances respectively. In each iteration, only one instance is queried. Training is always performed from scratch. In total, we perform 500 learning iterations. We perform our experiments with fully labelled datasets, which allows us to simulate manual labelling, a common practice in evaluating the performance of AL approaches (Dor et al., 2020; Lewis and Gale, 1994). For the Logistic Regression classifier, we rely on the default implementation of the Scikit-learn library[1]. We set the maximum number of iterations to 100. We use the FastText implementation provided by (Joulin et al., 2016). We set the embedding size to 10, train for 5 epochs and use a learning rate of 0.1. For the FastText classifier, we do not use the density of the learned sentence representation for the query strategy, since

[1] https://scikit-learn.org/

Table 2: Micro and macro F1-scores of the BERT Consumer after 300 and 500 iterations.

Strategy	App Store				Hate Speech				Reuters					
	FastText	SBERT	BERT	ELMo	FastText	SBERT	BERT	ELMo	FastText	SBERT	BERT	ELMo		
random	83.30	83.07	83.03	83.12	88.29	87.91	87.93	88.51	88.38	88.28	89.11	88.85	micro F1-score	300 Iterations
U	84.42	83.24	84.69	83.78	91.24	88.27	90.94	**93.02**	**91.33**	89.69	**92.15**	**91.43**		
U*DS	-	83.48	**84.59**	84.32	-	91.16	90.90	**92.27**	-	89.44	91.39	**92.33**		
IS_{LI}	83.15	82.49	84.02	83.26	91.13	89.36	**91.43**	92.03	86.30	91.11	**92.23**	90.59		
IS_{LI}*DS	-	83.50	**85.03**	81.93	-	87.64	91.50	91.87	-	90.45	**92.42**	**92.62**		
IS_{LS}	83.92	82.27	**85.13**	83.59	90.52	85.28	**91.85**	**92.45**	86.88	**90.98**	**92.82**	91.14		
IS_{LS}*DS	-	82.43	**84.95**	82.57	-	90.12	91.07	**92.56**	-	90.49	**93.16**	**91.54**		
AVG (unc)	83.83	82.78	84.74	83.24	90.96	88.64	91.28	92.37	88.18	90.36	92.36	91.61		
random	78.50	78.18	78.19	78.27	70.36	69.07	69.25	71.22	65.79	66.16	68.67	67.17	macro F1-score	
U	79.98	78.87	80.62	78.78	83.76	80.34	82.87	87.20	74.61	**80.25**	**79.99**	**79.25**		
U*DS	-	79.05	80.47	79.73	-	84.82	83.46	85.30	-	**80.36**	**79.06**	**79.71**		
IS_{LI}	78.14	77.55	79.43	78.55	84.21	80.65	83.77	85.10	63.00	**83.62**	**79.02**	72.37		
IS_{LI}*DS	-	79.11	**81.08**	77.22	-	78.17	83.18	85.03	-	**83.25**	**80.48**	**79.74**		
IS_{LS}	79.35	77.42	**81.08**	79.51	83.16	76.91	84.29	86.32	67.32	**82.96**	**79.56**	**79.52**		
IS_{LS}*DS	-	78.05	**80.84**	77.95	-	82.49	83.03	86.60	-	82.41	**80.28**	**80.00**		
AVG (unc)	79.16	78.12	80.59	78.62	83.71	80.56	83.44	85.93	68.31	82.14	79.73	78.43		
random	85.06	84.64	85.06	84.76	93.09	93.09	93.23	92.84	92.54	92.29	92.56	92.50	micro F1-score	500 Iterations
U	**86.29**	84.71	**86.55**	**86.23**	93.83	93.39	93.60	**94.35**	**94.56**	**95.41**	**95.38**	**95.19**		
U*DS	-	85.04	**86.38**	**86.08**	-	93.49	**94.30**	**94.62**	-	**95.39**	**95.60**	**95.44**		
IS_{LI}	85.96	85.27	**86.38**	**86.31**	**94.31**	92.77	**94.29**	**94.54**	91.21	**95.12**	**95.33**	**95.95**		
IS_{LI}*DS	-	85.47	86.27	85.06	-	93.58	**94.28**	**94.39**	-	**94.97**	**95.37**	**95.08**		
IS_{LS}	85.96	84.99	**86.23**	85.64	**94.29**	93.58	**93.98**	**94.51**	92.52	**95.13**	**94.90**	**95.65**		
IS_{LS}*DS	-	85.61	**86.45**	85.66	-	**93.98**	**93.94**	**94.64**	-	**95.37**	**95.32**	**95.37**		
AVG (unc)	86.07	85.18	86.38	85.83	94.14	93.47	94.06	94.51	92.76	95.23	95.32	95.45		
random	80.75	80.32	80.84	80.48	86.92	87.09	87.46	86.64	76.57	75.40	76.91	75.75	macro F1-score	
U	82.48	80.49	**82.89**	**82.41**	88.93	88.55	88.35	**89.95**	82.50	**91.32**	**87.18**	**87.01**		
U*DS	-	81.18	**82.59**	**82.11**	-	88.51	**89.61**	**90.25**	-	**91.04**	**88.68**	**88.78**		
IS_{LI}	82.27	81.32	**82.71**	**82.66**	**89.71**	87.45	**89.63**	**90.25**	77.49	**90.69**	**88.42**	**91.15**		
IS_{LI}*DS	-	81.66	82.29	80.87	-	88.95	**89.75**	**90.14**	-	**90.45**	**88.01**	**87.22**		
IS_{LS}	82.35	81.13	82.49	81.83	**89.87**	88.62	**89.12**	**90.08**	82.48	**90.31**	**84.80**	**88.40**		
IS_{LS}*DS	-	81.99	**82.77**	81.61	-	89.55	88.57	**90.65**	-	**91.42**	**87.55**	**87.35**		
AVG (unc)	82.37	81.30	82.62	81.92	89.50	88.61	89.17	90.22	80.82	90.87	87.44	88.32		

these are not available in our FastText implementation. We implement BERT using huggingface[2]. We use the *bert-base-uncased* pre-trained model and perform fine-tuning over 5 iterations. Further, we set $K = 20$ to estimate the density (Eq. 1) as proposed by the original authors. Further, we calculate the instability (Eq. 2, 3) across the $l = 5$ recent iterations. All experiments are performed on an Intel® Core™ i7-8550U CPU @ 1.80GHz with 16 GB RAM.

6 RESULTS

6.1 Performance of Proxy-Based AL (RQ1)

Table 2 lists the F1-scores of the BERT Consumers when trained via Proxy-based AL using 300 and 500 iterations. The table is organized in sampling strategies, classifiers, datasets, and number of AL iterations. The average (AVG unc) F1-scores across all selection strategies are stated at the bottom of each table. Significant improvements (paired t-test with p-value < 0.05) in regard to a random selection strategy are highlighted in bold. The best and worst performing embeddings for each dataset and selection strategy are marked as green and red, respectively.

Our results show that across all datasets, Proxy-based AL can significantly improve the F1-score of a target BERT classifier compared to a random selection strategy. After 500 iterations, a relative improvement of the micro F1-score of 1.75, 1.15, and 3.18% were obtained respectively for each dataset. The macro F1-score increased by 2.54, 2.62 and 15.30%. Across all experiments, BERT and ELMo embeddings provide the best Consumer performances. FastText provides the second best scores on the App Store dataset, but does not perform well on the Reuters dataset. SBERT embeddings perform worst and do only reach significant improvements on the Reuters dataset. The improvements in F1-score between the 300^{th} and 500^{th} iteration is straightforward, since more instances are used for training. Further, the macro F1-score improvements were much higher compared to the micro F1-score, which is caused by the highly unbalanced datasets. None of the strategies consistently outperforms the others, a common effect when evaluating AL (Dor et al., 2020). Overall, the results show that Proxy-based AL can significantly improve the F1-score compared to selecting instances

[2]https://huggingface.co/

Table 3: Relative F1-score improvements of Proxy-based AL (BERT as Consumer) compared to traditional AL (FastText and Logistic Regression) using 300 and 500 iterations.

Strategy	App Store				Hate Speech				Reuters					
	FastText	SBERT	BERT	ELMo	FastText	SBERT	BERT	ELMo	FastText	SBERT	BERT	ELMo		
U	1.34	1.37	5.31	1.97	3.34	-4.43	3.21	6.24	3.34	-6.50	1.62	-2.27	micro F1-score	300 Iterations
U*DS	-	2.19	5.70	2.01	-	-1.23	2.94	5.56	-	-6.75	0.76	-1.24		
IS_{LI}	-0.17	0.99	4.63	1.26	3.21	-3.15	4.14	4.89	-2.36	-5.00	2.53	-3.06		
IS_{LI}*DS	-	1.72	5.78	-0.37	-	-5.19	3.92	4.67	-	-5.64	2.88	-0.76		
IS_{LS}	0.74	0.14	6.57	1.17	2.53	-7.48	4.36	5.31	-1.70	-5.08	2.46	-2.65		
IS_{LS}*DS	-	-0.82	6.25	0.73	-	-2.34	3.40	5.57	-	-5.39	3.18	-2.01		
AVG (unc)	0.57	0.93	5.71	1.13	3.03	-3.97	3.66	5.37	-0.24	-5.73	2.24	-2.00		
U	1.88	2.82	8.25	1.73	19.05	-6.54	8.72	17.99	13.42	-12.44	-2.70	-9.24	macro F1-score	
U*DS	-	3.28	8.64	1.83	-	-1.13	8.21	16.23	-	-12.23	-3.96	-9.14		
IS_{LI}	-0.45	1.79	7.50	1.93	19.69	-5.83	10.14	12.66	-4.23	-8.56	2.06	-16.59		
IS_{LI}*DS	-	3.15	9.61	0.05	-	-9.05	8.30	13.15	-	-9.12	0.92	-8.74		
IS_{LS}	1.08	0.53	10.70	2.08	18.19	-10.33	10.80	15.28	2.33	-9.47	-2.68	-9.14		
IS_{LS}*DS	-	-0.70	10.01	1.42	-	-3.86	9.32	15.25	-	-9.49	-1.32	-8.09		
AVG (unc)	0.84	1.81	9.12	1.51	18.98	-6.12	9.25	15.09	3.84	-10.22	-1.97	-10.16		
U	1.45	2.44	6.59	3.66	0.80	0.72	5.55	6.82	2.19	-0.84	3.66	0.90	micro F1-score	500 Iterations
U*DS	-	2.81	6.73	3.67	-	0.76	6.06	7.27	-	-0.89	4.03	1.22		
IS_{LI}	1.03	2.89	6.58	3.52	1.31	0.15	6.49	6.86	-1.43	-1.10	3.66	1.73		
IS_{LI}*DS	-	2.79	7.00	2.01	-	0.68	6.30	6.74	-	-1.34	3.96	0.85		
IS_{LS}	1.07	2.33	6.12	2.71	1.29	0.93	6.09	7.26	-0.02	-1.29	3.25	1.40		
IS_{LS} *DS	-	2.95	6.76	2.79	-	1.29	5.95	6.95	-	-0.99	3.83	1.11		
AVG (unc)	1.18	2.70	6.63	3.06	1.13	0.75	6.07	6.98	0.24	-1.08	3.73	1.20		
U	2.14	3.80	9.80	4.63	2.32	2.36	13.82	18.61	7.75	-0.90	2.54	-2.22	macro F1-score	
U*DS	-	4.37	9.93	4.49	-	1.95	14.44	19.34	-	-1.27	5.05	-0.39		
IS_{LI}	1.88	4.52	10.11	4.95	3.22	1.25	15.57	16.71	1.21	-1.50	3.74	2.54		
IS_{LI}*DS	-	4.29	10.73	2.55	-	2.31	15.30	17.20	-	-1.99	4.53	-2.06		
IS_{LS}	1.98	3.77	9.81	3.67	3.40	2.41	15.43	18.73	7.72	-2.28	0.10	-0.66		
IS_{LS}*DS	-	4.70	10.29	3.95	-	3.40	14.85	17.84	-	-0.98	3.91	-1.98		
AVG (unc)	2.00	4.24	10.11	4.04	2.98	2.29	14.90	18.07	5.56	-1.49	3.31	-0.80		

at random, which would cause no waiting times at all.

6.2 Proxy-Based AL Compared to Traditional AL (RQ2)

The question arises whether it is worth to train an additional BERT classifier as performed in Proxy-based AL or whether a traditional AL approach would lead to similar F1-scores. Figure 3 shows the relative improvement of a FastText and Logistic Regression classifiers used in traditional AL compared to training an additional BERT classifier on the selected training set.

The results show that training a BERT Consumer can improve the micro F1-score up to 7.27% and the macro F1-score up to 19.34% compared to traditional AL.

Using less training data, i.e. 300 instances, only BERT, ELMo and sometimes FastText provide strong improvements ($> 3\%$) whereas SBERT performs similar or even worse than the Proxy alone. In contrast, using 500 instances as the training data for Proxy-based AL, both the App Store and the Hate Speech datasets show strong improvements. Only on the Reuters dataset, no improvements were reached compared to a stand-alone Proxy. Proxy-based AL profit from a larger number of iterations, since all F1-scores improved when performing 500 iterations compared to 300. While SBERT embeddings reach the highest F1-scores when applying self-selection, they provide the worst improvements within Proxy-based AL.

6.3 Run-Time of Proxy-Based AL (RQ3)

Finally, we investigate the time behaviour of Proxy-based AL. Rapid training, inference and data selection times are mandatory for Proxy-based AL to enable fast interaction cycles. Table 4 shows the time needed to perform the 500^{th} iteration across the algorithmic settings outlined in Section 4. The table shows the averaged run-time of all selection strategies. Averaging is performed to keep the figure clear, since no large differences between the selection strategies were observed. We perform all experiments on a CPU as outlined in Section 5.

All steps in the AL loop were carried out in less than 1 second using Logistic Regression with pre-trained text encodings. SBERT has shown to be the fastest approach, followed by BERT and lastly ELMo, which takes up to 0.68 seconds for the 500^{th} iteration. FastText is much slower, taking > 6 seconds for the 500^{th} iteration, which is too slow for real-time AL. Overall, the total runtime has shown to grow linear with respect to the number of iterations. We consider Proxy-based AL using Logistic Regression as fast enough to be performed in real-time applica-

Table 4: Run-time of the 500^{th} iteration in seconds on a CPU using Logistic Regression as the Proxy.

Runtime	App Store				Hate Speech				Reuters			
	FastText	SBERT	BERT	ELMo	FastText	SBERT	BERT	ELMo	FastText	SBERT	BERT	ELMo
Training	6.62	0.08	0.17	0.20	6.68	0.14	0.27	0.30	7.42	0.04	0.14	0.15
Inference	0.11	0.01	0.01	0.01	0.31	0.02	0.01	0.02	0.53	0.04	0.02	0.05
Selection	0.08	0.12	0.12	0.15	0.29	0.14	0.13	0.16	0.37	0.39	0.39	0.48
Total	6.81	0.21	0.30	0.36	7.28	0.29	0.41	0.49	8.32	0.47	0.54	0.68

tions without causing large interrupts. Further, the run-times indicate that a batch size of one is appropriate and there is no need to use batch-based selection strategies in order to save runtime even on a CPU.

7 DISCUSSION

The results demonstrate that Proxy-based AL can provide significant improvements in F1-score within a low-budget labelling setting. We found that our Proxy-based data selection increases the micro F1-score up to 7.27% and the macro F1-score up to 19.34% compared to randomly selecting instances using a state-of-the-art BERT classifier as the Consumer when labelling less than 500 instances manually. Furthermore, we show that Proxy-based AL can be used in real-time applications since it requires far less than 1 second for each iteration. We demonstrated that a Logistic Regression classifier is up to $\sim$ 33 times faster and provides a better reusability of the sampled data than FastText, which was previously considered as the state-of-the-art for rapid foreign selection (Coleman et al., 2020). We also show that Logistic Regression is a very strong baseline, and training a BERT model on the same training data might not lead to a better performance. Overall, Proxy-based AL does not increase the labelling effort compared to traditional AL while providing significantly better results and enabling rapid interaction cycles.

Table 5: BERTs' maximal reachable F1-score when using the entire data pool as training data.

F1-score	App Store	Hate Speech	Reuters
micro	87.82	96.19	97.17
macro	84.30	93.21	93.94

The maximum reachable F1-scores of BERT are shown in Table 5. The F1-scores are obtained when the entire data pool is labelled and used for training. The F1-scores show that on the Reuters dataset, the Proxy already reaches up to 99% of the maximum reachable F1-score after 500 iterations. In this case, training an additional BERT classifier provides no improvements compared to the original Proxy. In comparison, the Proxy reaches only 90-96% of the maximum F1-score on the App Store and Hate Speech datasets. Thus, Proxy-based AL is most beneficial when the Proxy's maximum reachable F1-score is much lower than that of BERT.

Interactively re-training classifiers has become an integral part of many real-world applications (Yarlagadda et al., 2021; Andersen et al., 2021). Typically, classifiers are frequently re-trained while humans are continuously interacting with a user interface. Proxy-based AL provides a framework to enable a rapid flow of user-interactions during the acquisition of training data while significantly improving the performance. Our findings also indicate that existing AL applications are likely to benefit from additionally training a state-of-the-art Consumer on the already acquired training data.

8 CONCLUSION

This paper investigates the potential of Proxy-based AL to maintain real-time learning within a low-budget labelling setting. We evaluate different algorithmic settings for Proxy-based AL including two classifiers, three datasets, seven data selection strategies and three text embeddings. We focus on BERT as the Consumer and FastText and Logistic Regression as the Proxy. We show that Proxy-based AL can improve the F1-score by 7.27 to 19.34% compared to traditional AL. Further, we demonstrate that a Logistic Regression Proxy is very fast, taking less than 1 second for each iteration and thus enabling interactive ML in real-time. Further work should investigate stop criteria for Proxy-based AL to efficiently spend human efforts. Also, the performance of Proxy-based AL on balanced datasets should be investigated, since we focus on unbalanced datasets.

REFERENCES

Andersen, J. S., Zukunft, O., and Maalej, W. (2021). Rem: Efficient semi-automated real-time moderation of online forums. In *In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 142–149.

Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., and Zaharia, M. (2020). Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations (ICLR)*.

Davidson, T., Warmsley, D., Macy, M., and Weber, I. (2017). Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 4171–4186.

Doherty, R. A. and Sorenson, P. (2015). Keeping users in the flow: mapping system responsiveness with user experience. *Procedia Manufacturing*, 3:4384–4391.

Dor, L. E., Halfon, A., Gera, A., Shnarch, E., Dankin, L., Choshen, L., Danilevsky, M., Aharonov, R., Katz, Y., and Slonim, N. (2020). Active learning for bert: An empirical study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962.

Fails, J. A. and Olsen Jr, D. R. (2003). Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 39–45.

Hu, R., Mac Namee, B., and Delany, S. J. (2016). Active learning for text classification with reusability. *Expert systems with applications*, 45:438–449.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. (2016). Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Joulin, A., Grave, É., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, volume 2, pages 427–431.

Lewis, D. D. and Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier.

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

Lewis, D. D., Yang, Y., Russell-Rose, T., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Lowell, D., Lipton, Z. C., and Wallace, B. C. (2019). Practical obstacles to deploying active learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 21–30.

Maalej, W., Kurtanović, Z., Nabil, H., and Stanik, C. (2016). On the automatic classification of app reviews. *Requirements Engineering*, 21(3):311–331.

Martin, G. L. and Corl, K. G. (1986). System response time effects on user productivity. *Behaviour & Information Technology*, 5(1):3–13.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Prabhu, A., Dognin, C., and Singh, M. (2019). Sampling bias in deep active classification: An empirical study. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4058–4068.

Pranckevičius, T. and Marcinkevičius, V. (2017). Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. *Baltic Journal of Modern Computing*, 5(2):221.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. (2020). Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Scheffer, T., Decomain, C., and Wrobel, S. (2001). Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer.

Settles, B. (2009). Active learning literature survey. In *ComputerSciences Technical Report 1648*. University of Wisconsin–Madison.

Settles, B. (2011). From theories to queries: Active learning in practice. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 1–18. JMLR.

Sugiyama, M. and Nakajima, S. (2009). Pool-based active learning in approximate linear regression. *Machine Learning*, 75(3):249–274.

Tolia, N., Andersen, D. G., and Satyanarayanan, M. (2006). Quantifying interactive user experience on thin clients. *Computer*, 39(3):46–52.

Tomanek, K. and Morik, K. (2011). Inspecting sample reusability for active learning. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 169–181. JMLR.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information retrieval*, 1(1):69–90.

Yarlagadda, S., Scroggins, D. J., Cao, F., Devabhaktuni, Y., Buitron, F., and Brown, E. T. (2021). Doctable: Table-oriented interactive machine learning for text corpora. In *2021 IEEE Workshop on Machine Learning from User Interactions (MLUI)*, pages 1–11. IEEE.

Zhu, J. and Ma, M. (2012). Uncertainty-based active learning with instability estimation for text classification. *ACM Transactions on Speech and Language Processing (TSLP)*, 8(4):1–21.

Zhu, J., Wang, H., Yao, T., and Tsou, B. K. (2008). Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling)*, pages 1137–1144.