DOMAIN ADAPTATION VIA ANOMALY DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Domain shift in finetuning from pre-training can significantly impact the performance of deep neural networks. In NLP, this led to domain specific models such as SciBERT, BioBERT, ClinicalBERT, and FinBERT; each pre-trained on a different, manually curated, domain-specific corpus. In this work, we present a novel domain-adaptation framework to tailor pre-training so as to reap the benefits of domain specific pre-training even if we do not have access to large domain specific pre-training corpus. The need for such a method is clear as it is infeasible to collect a large pre-training corpus for every possible domain. Our method is completely unsupervised and unlike related methods, works well in the setting where the target domain data is limited in size. We draw a connection between the task of adapting a large corpus to a target domain and that of anomaly detection, resulting in a scalable and efficient domain adaptation framework. We evaluate our framework and various baselines on eight tasks across four different domains: Biomedical, Computer Science, News, and Movie reviews. Our framework outperforms all the baseline methods and yields an average gain of 1.07% in performance. We also evaluate it on one of the GLUE task, sentiment analysis and achieve an improvement of 0.4% in accuracy.

1 INTRODUCTION

Pre-trained language models such as ELMo(Peters et al., 2018), GPT (Radford et al., 2018), BERT (Devlin et al., 2018), Transformer-xl (Dai et al., 2019) and XLNet (Yang et al., 2019) have become a key component in solving virtually all natural language tasks. BERT in particular has been the most popular such model in recent years. It achieves state-of-the-art performance on a wide variety of tasks such as sentiment analysis, question answering, semantic textual similarity and relation extraction. These models are pre-trained on large amount of cross-domain data ranging from Wikipedia to Book corpus to news articles, to learn powerful representations. A generic approach for using these models consists of two steps: (a) Pre-training: train the model on an extremely large general domain corpus with language modeling loss; (b) Finetuning: finetune the language model on labeled task dataset, for the downstream task.

Even though this approach has been very successful, it has a known weakness when applied to tasks containing text from a domain that is not sufficiently represented in the pre-training corpus. For instance, word distribution and their contextual meaning could be very different in the pre-training corpus and the task data resulting in limited applicability of embeddings learnt during pre-training. For this reason, the research community invested time and resources to pre-training language models on specific domains. The models include BioBERT pre-trained on biomedical text (Lee et al., 2020), ClinicalBERT pre-trained on clinical notes (Huang et al., 2019), SciBERT pre-trained on semantic scholar corpus (Beltagy et al., 2019), FinBERT pre-trained on financial documents (Araci, 2019). They achieve significant gain in performance when finetuned on tasks belonging to the same domain.

Despite its evident success, domain specific pre-training suffers from two issues when applied to new tasks. First, it requires explicit definition of the domain of the task data. Every task data has its own vocabulary, stylistic preferences and text characteristics which likely deviates from all encompassing general domain corpus used in pre-training. It may be hard to define a single genre for each new task and thus collect domain-specific data. The second, and likely more crucial issue is the cost of collecting domain-specific pre-training corpus. Pre-training is usually done on massive amount of data such as Wikipedia or Book corpus with millions of documents (several GBs of data) whereas the task data is usually very small (few MBs). Such task-domain specific corpus is hard

to find and very expensive to collect. For instance, clincalBERT used the public Multiparameter Intelligent Monitoring in Intensive Care (mimiciii) dataset (Johnson et al., 2016), that took long to be curated, then they had to parse it so that it could be processed by BERT. BioBERT is pre-trained on biomedical domain corpora PubMed. Authors had to crawl the documents in PubMed with abstracts having 4.5B words and the full-text articles with 13.5B words.

The question we attempt to answer in this paper is: *can the pre-training procedure on a generic large corpus be automatically adapted to a custom task domain?* Given an affirmative answer to this question, the major pain-points listed above are no longer there, and we would be able to boost the performance of language models on multiple domains, even if a large corpus of documents related to said domain cannot be found.

At a high level, our approach is motivated by the following observation; in a corpus that is rich and diverse enough, there must be many documents/sentences that are related to the domain of interest. Figure 1 shows an example where task domain data is identified in the general domain corpus data.

RCT20K TASK DATA	News Article
To investigate the efficacy of 6 weeks of daily low-dose oral prednisolone in improving pain, mobility, and systemic los-grade . . . [OBJECTIVE] A total of 125 patients with primary knee OA were randomize . . . [METHODS] Outcome measures included pain reduction and systemic inflammation markers . . . [METHODS]	For as much as we workout warriors recite that whole "no pain, no gain" mantra, we sure do pop a lot of painkillers. A recent article published in. . . These popular medicines, known as nonsteroidal anti-inflammatory drugs, or NSAIDs, work by suppressing inflammation. . . the article kind of blows past is the fact plenty of racers . . .

Figure 1: Identification of task-data (left panel, medical data) in general domain corpus (right panel).

One way to take advantage of this observation is to discover the subset of documents most similar to the task domain and run pre-training only on it. A softer approach would be to consider not an actual subset but a re-weighting of the instances in the pre-training set; this is of course a generalization of the subset approach. The main question with this approach is how to find the relevance weights in a way that is both reliable and scalable. Previous papers tried to solve this issue with either a simple language model (Moore & Lewis, 2010; Axelrod et al., 2011; Duh et al., 2013; Wang et al., 2017b; van der Wees et al., 2017), or by using a hand crafted similarity score (Wang et al., 2017a; Plank & Van Noord, 2011; Remus, 2012; Van Asch & Daelemans, 2010). The former works are on the one hand simplistic due to the nature of the simple language model they use and on the other hand require a fairly large corpus of task data in order to train a reasonable language model. The latter works have been shown not to generalize well to new tasks. We elaborate on these techniques in Section 2.

In this work, we propose *Domain adaptation via ANomAly detection* - DANA, a domain adaptation framework based on anomaly detection. DANA eliminates major limitations of the existing domain adaptation approaches. To handle scenarios wherein task data is too small to train a language model, we exploit pre-trained models to get a representation of the sentences. Further, instead of using these representations directly via some ad hoc techniques that do not generalize well to new tasks, we draw a connection to the area of anomaly detection. Indeed, in the problem of anomaly detection we must answer how likely is a new item to belong to a collection. The collection is the task data corpus, transformed into a vectorized representation using the pre-trained language model, and the new item is a candidate sentence from the large general domain corpus. By drawing this connection we are able to take advantage of well thought-out techniques, proven to work on a wide variety of domains, ensuring that our method generalizes well.

We present a comparative study of various anomaly detection methods for their usefulness in estimating relevance weights of sentences of the large general domain corpus. We establish a quantitative criterion and provide a data-driven approach to identify the best method for a given task data. Further, we provide two approaches for converting the raw anomaly scores into continuous/discrete relevance weights. To establish the performance gain of DANA, we evaluate it on eight tasks across four domains: Biomedical, Computer Science, News, and Movie reviews. We investigate all aspects of DANA by comparing its performance with various baselines based on its variants and the competitive methods available in literature. In particular, we compare DANA with language model

based domain adaptation technique proposed in Moore & Lewis (2010), distance scoring function based text filtration technique given in Wang et al. (2017a), and continued pre-training on the text data as done in Gururangan et al. (2020). DANA outperforms all these baseline methods and yield an average gain of 1.07% in accuracy.

2 Related Work

There has been a sequence of works in trying to find the relevance weights via language models Moore & Lewis (2010); Wang et al. (2017b); van der Wees et al. (2017). For instance, Moore & Lewis (2010), Axelrod et al. (2011) and Duh et al. (2013) train two language models, an in-domain language model on the target domain dataset (same as task domain in our case) and an out-of-domain language model on (a subset of) general domain corpus. Then, relevance score is defined as the difference in the cross-entropy w.r.t. two language models. These methods achieve some gain over the baseline methods but have two major drawbacks. First is the crucial assumption they rely on: *there is enough in-domain data to train a reasonable in-domain language model.* This assumption is not true in most cases. For most tasks, we only have access to a few thousands or in some cases a few hundreds of examples which is not enough to train a reasonably accurate language model. Second is that these methods fail to bridge the gap in performance created due to absence of large domain specific pre-training corpus. Models pre-trained on data filtered by these methods are significantly outperformed by models pre-trained on domain-specific pre-training corpus.

Another line of work defines hand crafted domain similarity measures to assign relevance score and filter out text from a general domain corpus (Wang et al., 2017a; Plank & Van Noord, 2011; Remus, 2012; Van Asch & Daelemans, 2010). For instance, Wang et al. (2017a) define the domain similarity of a sentence as the difference between Euclidean distance of the sentence embedding from the mean of in-domain sentence embeddings and the mean of out-of-domain sentence embeddings. Plank & Van Noord (2011) and Remus (2012) define the similarity measure as the Kullback-Leibler (KL) divergence between the relative frequencies of words, character tetra-grams, and topic models. Van Asch & Daelemans (2010) define domain similarity as Rényi divergence between the relevant token frequencies. These are adhoc measures suitable only for the respective tasks, and are a poor version of anomaly detection. They fail to generalize well for new tasks and domains. Ruder & Plank (2017) attempts to remedy this issue and tries to learn the correct combination of these metrics for each task. They learn the combination weight vector via Bayesian optimization. However, Bayesian optimization is infeasible for deep networks like BERT. Each optimization step of this process amounts to pre-training the model and finetuning it for the task. One needs to train the model many times which is infeasible in the realm of deep networks. Thus, they use models such as linear SVM classifier and LDA which do not yield state-of-the-art performance. In contrast, we propose a lightweight method - based on anomaly detection - that can be applied to state-of-the-art deep language models like BERT.

3 DANA: Domain adaptation via ANomALy Detection

Language model and Downstream Tasks. A generic approach for using state-of-the-art language models such as ELMo, GPT, BERT, and XLNet is to pre-train them on an extremely large general domain corpus and then finetune the pre-trained model on the downstream labeled task data. There is evident correlation between model's pre-training loss and its performance on the downstream task after finetuning (Devlin et al., 2018). Our design is motivated by an observation, backed by empirical evidence, that the correlation is even stronger if we consider the pre-training loss not on the pre-training data but the downstream task data.

To make this distinction formal, let $\mathcal{D}$, $\mathcal{D}_{\text{in}}$ be the pre-training and task data. Let Θ denote the parameters of the language model and ℓ_{LM} denote the language model loss function. The pre-training loss on pre-training data ($L_{\text{LM}}(\Theta)$) and target data ($L^{\text{in}}_{\text{LM}}(\Theta)$) are defined as follows

$$L_{\text{LM}}(\Theta) = \sum_{x \in \mathcal{D}} \ell_{\text{LM}}(x; \Theta), \quad L^{\text{in}}_{\text{LM}}(\Theta) = \sum_{x \in \mathcal{D}_{\text{in}}} \ell_{\text{LM}}(x; \Theta)\,.$$

To show that $L^{\text{in}}_{\text{LM}}(\Theta)$ is better correlated with the performance of the downstream task we consider several BERT language models pre-trained on random combinations of datasets from different do-

mains mentioned in Section 4. These models are selected such that for all the models $L_{\text{LM}}(\Theta)$ is roughly the same. For each model Θ, we estimate $L^{\text{in}}_{\text{LM}}(\Theta)$ and contrast it with the downstream accuracy/f1 score by finetuning the model on the labeled task data.

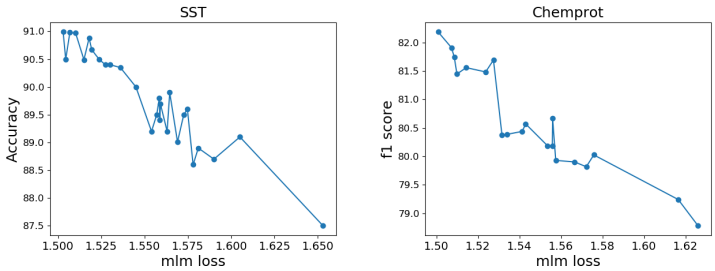

Figure 2: MLM loss of pre-trained BERT on the task data vs f1 score of corresponding finetuned BERT. Different points correspond to different BERT models, pre-trained on random combination of different datasets. Details of the tasks, SST and Chemprot can be found in Section 4.

Figure 2 provides the plots corresponding to this experiment and shows clear evidence that if the language model performs better on the task domain data, then the performance (accuracy/f1 score) of the finetuned model improves. We conclude that in order to ensure success in the downstream task, we should aim to minimize $L^{\text{in}}_{\text{LM}}(\Theta)$. A first attempt would be to pre-train or finetune the language model on $\mathcal{D}_{\text{in}}$. However, training a language model such as ELMo, GPT, BERT or XLNet requires a large corpus with several GBs of text and the available domain specific corpus $\mathcal{D}_{\text{in}}$ is often just the task data which has few MBs of text. Training on such a small dataset would introduce high variance. We reduce this variance by taking training examples from the general domain corpus $\mathcal{D}$, but control the bias this incurs by considering only elements having high relevance to the domain $\mathcal{D}_{\text{in}}$. Formally, we optimize a weighted pre-training loss function

$$L^{\Lambda}_{\text{LM}}(\Theta) \quad = \quad \sum_{x \in D} \lambda(x, \mathcal{D}_{\text{in}}) \cdot \ell_{\text{LM}}(x; \Theta)\,, \tag{1}$$

where $\lambda(x, \mathcal{D}_{\text{in}})$ are relevance weights of instance x for domain $\mathcal{D}_{\text{in}}$. $\lambda(x, \mathcal{D}_{\text{in}})$ is (close to) 1 if x is relevant to $\mathcal{D}_{\text{in}}$ and (close to) 0 otherwise. We compute these weights using an anomaly detection model fitted on $\mathcal{D}_{\text{in}}$.

3.1 Anomaly Detection to solve the Domain Membership Problem

Detecting whether an instance x is an in-domain instance is equivalent to solving the following problem: *Given task data $\mathcal{T}$ and a sentence s, determine if s is likely to come from the distribution generating $\mathcal{T}$ or if s is an anomaly.*

This view helps us make use of a wide variety of anomaly detection techniques developed in literature (Noble & Cook, 2003; Chandola et al., 2009; Chalapathy & Chawla, 2019). To make use of these techniques, we first need a good numeric representation (embedding) with domain discrimination property. We use pre-trained BERT to embed each sentence into a 768 dimensional vector. Once the data is embedded, we need to decide which among the many anomaly detection algorithms proposed in literature should be applied on the embeddings. To decide the anomaly detection method, we propose an evaluation method ranking the techniques based on their discriminative properties.

Ranking anomaly detection algorithms: The idea is to treat the anomaly score as the prediction of a classifier distinguishing between in-domain and out-of-domain data. By doing so, we can consider classification metrics such as the f1_score as the score used to rank the anomaly detection algorithm. To do this, we split the in-domain data (the task data) into $\mathcal{D}^{\text{train}}_{\text{in}}, \mathcal{D}^{\text{test}}_{\text{in}}$ using a 90/10 split. We also create out-of-domain data $\mathcal{D}_{\text{out}}$ as a random subset of $\mathcal{D}$ of the same size as $\mathcal{D}^{\text{test}}_{\text{in}}$. We train an anomaly detection algorithm A with $\mathcal{D}^{\text{train}}_{\text{in}}$, and evaluate it's f1_score on the labeled test set composed of the union $\mathcal{D}^{\text{test}}_{\text{in}} \cup \mathcal{D}_{\text{out}}$, where the labels indicate which set the instance originated from.

Table 1 provides the results of this evaluation on six anomaly detection algorithms. Details of the tasks can be found in Section 4. We can see that Isolation Forest consistently performs well for most of the tasks. Local Outlier Factor performs almost equally well but is slower in prediction. Although

it is possible to adaptively choose for every task the anomaly detection algorithm maximizing the f1_score, we chose to use a single algorithm, Isolation Forests, for the sake of having a simpler technique and generalizable results.

Task	RC	kNN	PCA	OCS	LOF	IF
CHEMPROT	0.89	0.85	0.92	0.87	0.92	**0.96**
ACL-ARC	0.77	0.88	0.90	0.89	**0.91**	0.88
HYPERPARTISAN	0.86	0.86	0.95	0.98	0.91	**0.98**
RCT20K	0.85	0.88	0.82	0.76	0.87	**0.93**
IMDB	0.88	0.96	0.87	0.81	**0.96**	0.94
SCIERC	0.78	0.84	0.86	0.76	0.88	**0.92**
HELPFULNESS	0.82	0.89	0.83	0.76	0.83	**0.92**
IMDB	0.84	0.89	0.80	0.73	**0.92**	0.87

Table 1: Scores of different anomaly detection algorithms for different tasks. RC: Robust Covariance (Nguyen & Welsch, 2010), kNN: Nearest neighbor (Gu et al., 2019), PCA: Principal Component Analysis (Harrou et al., 2015), OCS: One Class SVM (Schölkopf et al., 2000), LOF: Local Outlier Factor (Breunig et al., 2000), IF: Isolation Forest (Liu et al., 2008)

Isolation Forest (Liu et al., 2008): For completeness, we provide a brief description of the Isolation Forest algorithm. Isolation Forest is an unsupervised decision tree ensemble method that identifies anomalies by isolating outliers of the data. It isolates anomalies in data points instead of profiling the normal points. Algorithm works by recursively partitioning the data using a random split between the minimum and maximum value of a random feature. It works due to the observation that outliers are less frequent than the normal points and lie further away from normal points in the feature space. Thus, in a random partitioning, anomalous points would require fewer splits on features resulting in shorter paths and distinguishing from the rest of the points. Anomaly score of a point x is defined as $s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$, where $E[h(x)]$ is the expected path length of x in various decision trees, $c(n) = 2H(n-1) - 2(n-1)/n$ is the average path length of unsuccessful search in a Binary Tree and $H(n-1)$ is the $n-1$-th harmonic number and n is the number of external nodes.

Now that we chose the anomaly detection technique, we move to discuss the effectiveness of the algorithm in (i) identifying the domain from the task data (ii) identifying the domain related data from the general domain corpus. Figure 3 (left) shows that the anomaly detection algorithm is able

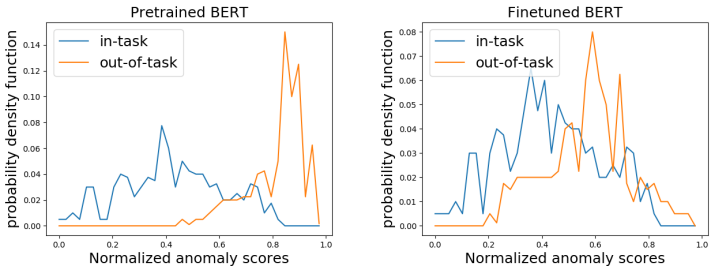

Figure 3: Sentence anomaly scores for SST with anomaly detection algorithm trained on embeddings from Left: pre-trained BERT, Right: finetuned BERT. In-task: sentences from the task data, out-of-task: sentences from general domain corpus.

to distinguish between the in-task-domain data and the out-of-task domain data. These experiments are done for the Sentiment Analysis task (SST) discussed in Section 4. Interestingly, we noticed in our experiments that a language model pre-trained on a diverse corpus is a better choice when compared to a model finetuned on the target domain. We conjecture that the reason is that a finetuned BERT is overly focused on the variations in the task data which are useful for task prediction and forgets information pertaining to different domains which is useful for domain discrimination. We exhibit this phenomenon more clearly in Figure 3 (right) where it is evident that the discriminating ability of the finetuned model is worse.

.

Corpus	Input data	Filtered (Bio)	Filtered (CS)
News	7.1GB (21.8%)	0.13GB (2.04%)	0.04GB (0.71%)
Finance	4.9GB (15.0%)	0.01GB (0.15%)	0.02GB (0.35%)
CS abstracts	8.0GB (24.5%)	1.00GB (15.41%)	5.08GB (78.01%)
Bio abstracts	12.6GB (38.7%)	5.37GB (82.4%)	1.36GB (20.94%)

Table 2: Filtering algorithm trained with Bio Abstracts and CS task data. We mix four corpora, filter out 80% of the data and retain the remaining 20% in both cases.

In order to assess the ability of our model to identify related text we perform the following experiment. First, we create a diverse corpus by taking the union of 4 datasets: News, Finance, CS abstracts and Biology abstracts. Table 2, column 'Input data' contains their respective sizes. We then train two anomaly score based discriminators, one on CS task data and the other on Bio abstracts. For each model we choose a threshold that would filter out 80% of the data, and observe the data eventually retained by it. The fraction of data retained from each corpus for each model is given in Table 2, columns 'Filtered (Bio)' and 'Filtered (CS)'. We see that data from the News and Finance corpus is almost completely filtered as it is quite different than the text in abstracts of academic papers. We also see that a non-negligible percent of the filtered data for the Bio model comes from CS and vice versa. Since both corpora are abstracts of academic papers it makes sense that each corpus contains relevant data for the other. The details related to the datasets used to construct the corpora above are given in Appendix A.

3.2 From Anomaly Detection Scores to Domain Adapted Pre-training

Once the anomaly detection object is trained, we use it to compute the relevance weights i.e. compute λ values defined in equation 1 . Let the sentences in the pre-training corpus be $s_1, \ldots, s_N$ with anomaly scores $\{A(s_1), \ldots, A(s_N)\}$. We explore two different strategies of λ value computation. First is when we normalize and transform the scores to compute continuous values and second when we use threshold and compute $0/1$ values.

Continuous λ values: We start by normalizing the anomaly scores to be mean zero and variance 1. Let $\mu = (\sum_{i=1}^{N} A(s_i))/N, \sigma = \sqrt{(\sum_{i=1}^{N}(A(s_i) - \mu)^2)/N}$. Then, for every $i \in \{1, \ldots, N\}$, normalized score is $\bar{A}(s_i) = (A(s_i) - \mu)/\sigma$. Using these normalized sentence anomaly scores, we compute the relevance weights as follows:

$$\lambda(s_i) = \frac{1}{1 + e^{-C(\alpha - \bar{A}(s_i))}}$$

where C and α are hyper-parameters. C controls the sensitivity of the weight in terms of anomaly score and α controls the fraction of target domain data present in the general domain corpus. $C \to \infty$ results in $0/1$ weights corresponding to discrete λ setting whereas $C = 0$ results in uniform λ values resulting in the no domain adaptation setting.

Discrete λ values: We sort the sentences as per anomaly scores, $A(s_{\sigma(1)}) \leq A(s_{\sigma(2)}) \leq \cdots \leq A(s_{\sigma(N)})$ and pick β fraction of the sentences with lowest anomaly scores,

$$\lambda(s_{\sigma(i)}) = 1 \text{ for } i \in \{1, \ldots, \beta N\} \text{ and } 0 \text{ otherwise}$$

Even though this approach is less general than the continuous λ values case, it has an advantage of being model independent. We can filter out text, save it and use it to train any language model in a black box fashion. It does not require any change in pre-training or finetuning procedure. However, to utilize this option we need to make a change. Instead of filtering out sentences we need to filter out segments containing several consecutive sentences.

To understand why, suppose we filter out sentence 1 and sentence 10 and none of the sentences in between. When we save the text and construct input instances from it for a language model, then an input instance may contain the end of sentence 1 and the start of sentence 10. This is problematic as sentence 1 and sentence 10 were not adjacent to each other in the original corpus and hence, language model does not apply to them. It distorts the training procedure resulting in worse language models. To resolve this issue, we group sentences into segments and classify the

relevance of each segment. Formally, let γ be a hyper-parameter and for all $j \in 1, \ldots, \lfloor N/\gamma \rfloor$ let the segment score be $y_j = \sum_{i=(j-1)*\gamma+1}^{j*\gamma} \frac{A(s_i)}{\gamma}$. We sort the segments according to their anomaly scores, $y_{\sigma'(1)} \leq \cdots \leq y_{\sigma'(N/\gamma)}$ and select the β fraction with lowest anomaly scores; save the sentences corresponding to these segments.

To completely avoid the issue, we may set segment length very large. However, this is not feasible as the diverse nature of pre-training corpus makes sure that large enough segments rarely belong to a specific domain, meaning that the extracted data will no longer represent our target domain. We experimented with a handful of options for the segment length, and found the results to be stable when choosing segments of 15 sentences.

Continued pre-training instead of pre-training from scratch: Once we have computed the relevance weights $\lambda(s_i)$, we do not start pre-training the language model from scratch as this is not feasible for each new task/domain. Instead, we start with a language model pre-trained on the general domain corpus and perform additional pre-training for relatively fewer steps with the weighted loss function. In our case, we start with a BERT language model pre-trained for one million steps and continued pre-training with updated loss function for either $50,000$ or $100,000$ steps.

4 EXPERIMENTS

Task	Train	Dev	Test	C	Task	Train	Dev	Test	C
CHEMPROT	4169	2427	3469	13	IMDB	20000	5000	25000	2
RCT20K	180040	30212	30135	5	ACL-ARC	1688	114	139	6
HYPERPARTISAN	516	64	65	2	SCIERC	3219	455	974	7
AGNEWS	115000	5000	7600	4	SST	67349	872	1821	2
HELPFULNESS	115251	5000	25000	2					

Table 3: Specification of task datasets. C refers to the number of classes. CHEMPROT (Kringelum et al., 2016) and RCT20K (Dernoncourt & Lee, 2017) are from biomedical domain. HYPERPARTISAN (Kiesel et al., 2019) and AGNEWS (Zhang et al., 2015) are from news domain. HELPFULNESS (McAuley et al., 2015) and IMDB (Maas et al., 2011) are from reviews domain. ACL-ARC (Jurgens et al., 2018) and SCIERC (Luan et al., 2018) are from CS domain. SST (Socher et al., 2013) is a general domain sentiment analysis task.

We use datasets listed in Table 3 along with a general domain corpus consisting of 8GB of text from Wikipedia articles. As a base for our experiments we use the $\text{BERT}_{\text{BASE}}$ model provided in the Gluon-NLP package. It has 12 layers, 768 hidden dimensions per token, 12 attention heads and a total of 110 million parameters. It is pre-trained with sum of two objectives. First is the masked language model objective where model learns to predict masked tokens. Second is the next sentence prediction objective where sentence learns to predict if sentence B follows sentence A or not. We use learning rate of 0.0001, batch size 256 and warm-up ratio 0.01. For finetuning, we pass the final layer [CLS] token embedding through a task specific feed-forward layer for prediction. We use learning rate 3e-5, batch size 8, warm-up ratio 0.1 and train the network for five epochs. In all the experiments, we start with a BERT pre-trained for one million steps and continue pre-training for $50,000$ steps in case of discrete λ case and $100,000$ steps in case of continuous λ case. Also, as mentioned in Section 3.2, we filter out segments instead of sentences and save them. We set the segment length to be 15 sentences and filter out 20% of the data. Pseudo-code of the end-to-end algorithm can be found in Appendix A.

4.1 BASELINE METHODS

For each baseline, we start with a BERT pre-trained on general domain corpus for one million steps as in case of our anomaly detection based method. Then, we continue pre-training the baseline method for the same number of steps as in case of our method. In case of baseline methods which filter general domain corpus, we filter the same fraction of text as in case of our method.

General: *Continued pre-training on general domain corpus*. We know that more pre-training leads to a better model. To estimate the impact of extra pre-training, we consider a baseline where we continue pre-training on the general domain corpus.

Random: *Continued pre-training on random subset of general domain corpus.*

Task (Gururangan et al., 2020): *Continued pre-training on task data*. We continue pre-training on the task text. Since task text is low, we can not pre-train on task data for as many steps as in other case. Instead we do 100 epochs, save the model after every 10 epoch and pick the best one.

LM (Moore & Lewis, 2010): *Continued pre-training on text filtered via language models trained on task data.* We train two language models, one on the task text and one on a subset of general domain corpus (same size as the task text). We select sentences with lowest scores given by the function $f(s) = H_I(s) - H_O(s)$ where $H_I(s)$ and $H_O(s)$ are the cross-entropy between the n-gram distribution and the language model distribution. More formally, cross entropy of a string s with empirical n-gram distribution p given a language model q_I is $H_I(s) = -\sum_x p(x) \log q_I(x)$.

Distance (Wang et al., 2017a): *Continued pre-training on text filtered via Euclidean distance scoring function.* For each sentence f, we consider BERT embedding v_f and compute vector centers $C_{F_{\text{in}}}$ and $C_{F_{\text{out}}}$ of the task data F_{in} and a random subset of general domain corpus F_{out}; $C_{F_{\text{in}}} = \frac{\sum_{f \in F_{\text{in}}} v_f}{|F_{\text{in}}|}, C_{F_{\text{out}}} = \frac{\sum_{f \in F_{\text{out}}} v_f}{|F_{\text{out}}|}$. We score a sentence f as per the scoring function:$\delta_f = d(v_f, C_{F_{\text{in}}}) - d(v_f, C_{F_{\text{out}}})$. We pick the text with lowest scores.

4.2 RESULTS

Task	Base	General	Random	Task	LM	Distance	DANA
CHEMPROT	$81.62_{0.74}$	$81.59_{0.67}$	$81.62_{0.71}$	$81.63_{0.82}$	$81.83_{0.74}$	$81.64_{0.76}$	$\mathbf{82.41}_{0.62}$
RCT20K	$87.52_{0.16}$	$87.57_{0.16}$	$87.54_{0.17}$	$87.60_{0.18}$	$\mathbf{87.85}_{0.23}$	$87.62_{0.24}$	$87.82_{0.13}$
HPRPARTISAN	$70.57_{3.04}$	$70.97_{2.03}$	$71.04_{2.32}$	$70.88_{2.63}$	$71.47_{2.56}$	$72.16_{2.14}$	$\mathbf{73.58}_{2.39}$
AGNEWS	$93.99_{0.13}$	$94.06_{0.19}$	$\mathbf{94.09}_{0.11}$	$94.04_{0.08}$	$94.03_{0.13}$	$94.04_{0.11}$	$94.03_{0.16}$
HELPFULNESS	$69.30_{0.60}$	$69.39_{0.78}$	$69.34_{0.58}$	$69.41_{0.50}$	$69.58_{0.59}$	$69.42_{0.69}$	$\mathbf{69.70}_{0.92}$
IMDB	$88.65_{0.24}$	$88.53_{0.27}$	$88.63_{0.26}$	$88.77_{0.39}$	$88.67_{0.44}$	$88.69_{0.47}$	$\mathbf{89.29}_{0.22}$
ACL-ARC	$72.31_{4.7}$	$72.38_{3.93}$	$72.42_{3.71}$	$72.46_{3.48}$	$72.40_{1.85}$	$72.47_{2.64}$	$\mathbf{72.81}_{3.83}$
SCIERC	$82.84_{1.39}$	$82.85_{1.38}$	$82.81_{1.13}$	$83.18_{1.09}$	$82.99_{2.75}$	$83.40_{2.17}$	$\mathbf{85.85}_{0.95}$
SST	$92.02_{0.29}$	$92.21_{0.31}$	$92.14_{0.24}$	$92.21_{0.24}$	$92.25_{0.4}$	$92.15_{0.35}$	$\mathbf{92.42}_{0.32}$

Table 4: Performance of DANA and five Baseline methods. Base corresponds to the pre-trained model on general domain corpus with no further pre-training. Baseline methods are mentioned in previous subsection. DANA corresponds to our method with discrete relevance weights. Keeping in line with the previous works, we use the following metrics: accuracy for SST, micro f1 score for CHEMPROT and RCT20K, macro f1 score for ACL-ARC, SCIERC, HELPFULNESS, HPRPARTISAN, IMDB, and AGNEWS. Each model is finetuned eight times with different seeds and the mean value is reported. Subscript correspond to the standard deviation in the finetuned model performance.

Table 4 shows the effectiveness of automatically adapting pre-training to the task domain. Our method achieves performance gain in all domains with an average gain of 1.01% over the base model pre-trained on the general domain corpus and beats all the baseline methods. Our method has the advantage that it does not require access to any large domain specific corpus. Instead we only have access to a very small task dataset available for finetuning. So, it is applicable to any new task from any new domain. Results are presented for discrete relevant weight case as they are better when the number of steps available to continue pre-training are small. Results for continuous relevance weights case can be found in Appendix C. We also observe that the performance boost is higher if the corresponding boost via additional pre-training on large domain specific corpus is higher. Results for this comparison can be found in Appendix D.

5 CONCLUSION

Recent development of various domain specific models in language modeling shows that domain shift in finetuning from pre-training can significantly deteriorate the performance of the downstream task. The existing domain adaptation methods either require sufficiently large task data, or are based on adhoc techniques that do not generalize well across tasks. Our major contribution is providing a new domain adaptation technique that performs well even with very little task data, and generalizes well across tasks.

REFERENCES

Waleed Ammar, Dirk Groeneveld, Chandra Bhagavatula, Iz Beltagy, Miles Crawford, Doug Downey, Jason Dunkelberger, Ahmed Elgohary, Sergey Feldman, Vu Ha, et al. Construction of the literature graph in semantic scholar. *arXiv preprint arXiv:1805.02262*, 2018.

Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*, 2019.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 355–362, 2011.

Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Franck Dernoncourt and Ji Young Lee. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. *arXiv preprint arXiv:1710.06071*, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 678–683, 2013.

Xiaoyi Gu, Leman Akoglu, and Alessandro Rinaldo. Statistical analysis of nearest neighbor methods for anomaly detection. In *Advances in Neural Information Processing Systems*, pp. 10923–10933, 2019.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

Fouzi Harrou, Farid Kadri, Sondes Chaabane, Christian Tahon, and Ying Sun. Improved principal component analysis for anomaly detection: Application to an emergency department. *Computers & Industrial Engineering*, 88:63–77, 2015.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*, 2019.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. Measuring the evolution of a scientific field through citation frames. *Transactions of the Association for Computational Linguistics*, 6:391–406, 2018.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. Semeval-2019 task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pp. 829–839, 2019.

Jens Kringelum, Sonny Kim Kjaerulff, Søren Brunak, Ole Lund, Tudor I Oprea, and Olivier Taboureau. Chemprot-3.0: a global chemical biology diseases mapping. *Database*, 2016, 2016.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422. IEEE, 2008.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. *arXiv preprint arXiv:1808.09602*, 2018.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pp. 43–52, 2015.

Robert C. Moore and William Lewis. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pp. 220–224, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P10-2041`.

Tri-Dzung Nguyen and Roy E Welsch. Outlier detection and robust covariance estimation using mathematical programming. *Advances in data analysis and classification*, 4(4):301–334, 2010.

Caleb C Noble and Diane J Cook. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 631–636, 2003.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Barbara Plank and Gertjan Van Noord. Effective measures of domain similarity for parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 1566–1576, 2011.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

Robert Remus. Domain adaptation using domain similarity-and domain complexity-based instance selection for cross-domain sentiment analysis. In *2012 IEEE 12th international conference on data mining workshops*, pp. 717–723. IEEE, 2012.

Sebastian Ruder and Barbara Plank. Learning to select data for transfer learning with bayesian optimization. *arXiv preprint arXiv:1707.05246*, 2017.

Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pp. 582–588, 2000.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Vincent Van Asch and Walter Daelemans. Using domain similarity for performance estimation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pp. 31–36, 2010.

Marlies van der Wees, Arianna Bisazza, and Christof Monz. Dynamic data selection for neural machine translation. *arXiv preprint arXiv:1708.00712*, 2017.

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 560–566, 2017a.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1482–1488, 2017b.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pp. 5753–5763, 2019.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, pp. 9054–9065, 2019.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pp. 649–657, 2015.

APPENDIX

A DATASETS IN ACCURACY ESTIMATION OF ANOMALY SCORE BASED DATA FILTRATION

CS task data: To train anomaly score discriminator for CS data, we use the tasks data from ACL-ARC and SCIERC. Details of these datasets are mentioned in Section 4.

CS and Bio Abstracts: Semantic Scholar corpus (Ammar et al., 2018) contains datasets from a variety of domain. We filter out text based on the domain field and only keep the abstracts from CS and bio domain.

News: We use REALNEWS (Zellers et al., 2019) corpus containing news articles from 500 news domains indexed by Google News. It is obtained by scraping dumps from Common Crawl.

Finance: We use the TRC2-financial dataset. This a subset of Reuters TRC24 corpus containing news articles published between 2008 and 2010. It can be obtained by applying here: `https://trec.nist.gov/data/reuters/reuters.html`

B PSEUDO CODE

Algorithm 1 Task Adaptive Pre-training

Input: Pre-trained model B, Pre-training instances $x_1, \ldots, x_N$, task data $\mathcal{T}$, (C, α), #steps
Stage 1: Instance weight computation
Let the sentences of the task be $s_1, \ldots, s_t$ with sentence embeddings $P = \{\text{Embed}(s_1), \ldots, \text{Embed}(s_t)\}$.
Let a random subset of pre-training instances (sentences of these instances) be $s'_1, \ldots, s'_{t/10}$ with BERT based sentence embeddings $N = \{\text{Embed}(s'_1), \ldots, \text{Embed}(s'_{t/10})\}$
Train an anomaly detection object, IF = IsolationForest$(P \cup N)$
For $i \in [N]$, let $S(x_i) = \text{IF.score}(\text{Embed}(x_i))$
Let $\mu = \frac{1}{N}\sum_{i=1}^{N} S(x_i)$ and $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(S(x_i) - \mu)^2}$.
For every $i \in [N]$, $\bar{S}(x_i) = \frac{S(x_i)-\mu}{\sigma}$.
For every $i \in [N]$, $\lambda(x_i) = \frac{1}{1+e^{-C(\alpha - \bar{S}(x_i))}}$
Stage 2: Adaptation of pre-training to target domain
Continue training language model B for #steps on instances $x_1, \ldots, x_N$ with instance weights $\lambda(x_1), \ldots, \lambda(x_N)$.
Finetune resulting model on the labeled task data $\mathcal{T}$

Algorithm 1 shows the pseudo code for the case of continuous relevance weights. Discrete relevance weight setting is same as $C \to \infty$. As discussed in 3.2, in case of discrete relevance weights, we filter out segments containing several consecutive sentences. We experimented with several options for the segment length and found the stable segment length to be 15 sentences. Here, a sentence is a consecutive piece of text such that when applied through the BERT tokenizer, it results in 256 sentences.

C CONTINUOUS RELEVANCE WEIGHTS

We see in Table 5 that a model additionally pre-trained for 50,000 with discrete λ values consistently over performs the continuous case even when we train with continuous relevance weights for far higher number of steps. This is because of the fact that many of those steps yield virtually no training at all. For instance, suppose the relevance weights are uniformly distributed between 0 and 1; $[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$. Then, in discrete case we pick the top two sentences and thus two steps are sufficient to train on these most relevant sentences (assume batch size is 1). However, in continuous case, we need to train the model for ten steps to train on these top two

Task	Base	Discrete	Continuous-1	Continuous-3
CHEMPROT	$81.62_{0.74}$	$\mathbf{82.41}_{0.62}$	$81.74_{0.81}$	$81.64_{0.83}$
RCT20K	$87.52_{0.16}$	$\mathbf{87.82}_{0.13}$	$87.49_{0.28}$	$87.56_{0.22}$
HPRPARTISAN	$70.57_{3.04}$	$\mathbf{73.58}_{2.39}$	$70.94_{1.98}$	$71.29_{2.95}$
AGNEWS	$93.99_{0.13}$	$\mathbf{94.03}_{0.16}$	$94.01_{0.14}$	$94.01_{0.15}$
HELPFULNESS	$69.30_{0.60}$	$\mathbf{69.70}_{0.92}$	$69.35_{0.5}$	$69.37_{0.44}$
IMDB	$88.65_{0.24}$	$\mathbf{89.29}_{0.22}$	$88.63_{0.51}$	$88.71_{0.46}$
ACL-ARC	$72.31_{4.7}$	$\mathbf{72.81}_{3.83}$	$72.26_{2.33}$	$72.36_{2.12}$
SCIERC	$82.84_{1.39}$	$\mathbf{85.85}_{0.95}$	$83.14_{1.96}$	$83.13_{2.65}$
SST	$92.02_{0.29}$	$\mathbf{92.42}_{0.32}$	$92.11_{0.32}$	$92.13_{0.37}$

Table 5: Comparison of discrete vs continuous relevance weight setting. Base corresponds to the pre-trained model on general domain corpus with no further pre-training. Discrete refers to DANA with discrete relevance weights/filtered out text and pre-trained additionally for 50000 steps. Continuous-x refers to DANA with continuous relevance weights and pre-trained additionally for $x*100,000$ more steps. Metrics used for different tasks: accuracy for SST, micro f1 score for CHEMPROT and RCT20K, macro f1 score for ACL-ARC, SCIERC, HELPFULNESS, HPRPARTISAN, IMDB, and AGNEWS. Each model is finetuned eight times with different seeds and the mean value is reported. Subscript correspond to the standard deviation in the finetuned model performance.

relevant sentences. Thus, we need many more steps to achieve and beat the performance achieved in the Discrete case. An open question is to combine the two settings so as to benefit from the generality of Continuous case and efficiency of the discrete case.

D PERFORMANCE BOOST WITH DOMAIN-SPECIFIC CORPUS VS DANA

Task	DANA	Domain Corpus
CHEMPROT	0.79	2.3
RCT20K	0.3	0.4
HYPERPARTISAN	3.01	1.6
AGNEWS	0.04	0.0
HELPFULNESS	0.4	1.4
IMDB	0.64	5.4
ACL-ARC	0.5	3.5
SCIERC	3.01	12.4

Table 6: Performance boost via DANA vs pre-training on domain specific corpus. DANA corresponds to our method with discrete relevance weights/filtered out text and pre-trained additionally for 50000 steps. Domain Corpus refers to the model trained in Gururangan et al. (2020) over the domain same as the downstream task. Metrics used for different tasks: accuracy for SST, micro f1 score for CHEMPROT and RCT20K, macro f1 score for ACL-ARC, SCIERC, HELPFULNESS, HPRPARTISAN, IMDB, and AGNEWS. Each model is finetuned eight times with different seeds. We report the difference in the mean value of performance between the model with additional pre-training and base model with no additional pre-training.

We compare the performance boost we achieved due to DANA with the performance boost we achieve if we have access to large pre-training corpus. In Table 6, we list the gain in performance in both cases over eight tasks from four domains. We see that the performance boost is higher with DANA if the corresponding boost is higher with domain specific corpus. Thus if there is a large domain shift between the general domain corpus and the task data, as can be measured by the performance boost via large pre-training corpus, then DANA is able to achieve large performance boost via Domain Adaptation. Scale of numbers in the two columns are not directly comparable due to the following two reasons. First is that additional pre-training done is Gururangan et al. (2020) is for almost as many steps as the number of steps required to pre-train a network from scratch. However, in our case additional pre-training is done for only 5% of the number of steps required to pre-train a network from scratch. Second reason is that the model used in (Gururangan et al., 2020) is different, ROBERTA. Also, the general domain corpus is different and thus the domain shift is not exactly the same as in our case. The point however remains the same, which is that as the target domain is further away from the pre-training corpus, the benefits of DANA increase.