Can Data Diversity Enhance Learning Generalization?

Yu Yu* and Shahram Khadivi [1] and Jia Xu*

* School of Engineering and Science, Steven Institute of Technology, NJ 07030, USA
[1] eBay Inc., Aachen 52064, Germany
yyu50@stevens.edu, skhadivi@ebay.com, jxu70@stevens.edu

Abstract

This paper introduces our Diversity Advanced Actor-Critic reinforcement learning (A2C) framework (DAAC) to improve the generalization and accuracy of Natural Language Processing (NLP). We show that the diversification of training samples alleviates overfitting and improves model generalization and accuracy. We quantify diversity on a set of samples using the max dispersion, convex hull volume, and graph entropy based on sentence embeddings in high-dimensional metric space. We also introduce A2C to select such a diversified training subset efficiently. Our experiments achieve up to +23.8 accuracy increase (38.0% relatively) in sentiment analysis, -44.7 perplexity decrease (37.9% relatively) in language modeling, and consistent improvements in named entity recognition over various domains. In particular, our method outperforms both domain adaptation and generalization baselines without using any target domain knowledge.

1 Introduction

We introduce the Diversity Advanced Actor-Critic reinforcement learning framework (DAAC) to improve the generalization and accuracy of Natural Language Processing (NLP). Training data plays a crucial role in Deep Learning (DL)-based NLP. Investigations (Hendrycks et al., 2020; Dodge et al., 2020; Ramponi and Plank, 2020) show that training data quality is imperative to the machine learning model’s performance. Good data are effective and easy to generalize, while bad data bring noise and overfits on out-of-domain test sets.

Decades of work (van der Wees et al., 2017; Aharoni and Goldberg, 2020; Guo et al., 2020; Axelrod et al., 2011) have been devoted to finding the best data, with the most commonly used method we call domain adaptation (Qu et al., 2019; Liu et al., 2019) that uses target domain knowledge. Despite large improvements on specific domains, these adapted models lack robustness and are prone to rare events. For example, when models shift towards one target domain like Twitter, they may become erroneous for other domains like medical or travel, and real-world applications on news report data, for example, do not know which queries to receive before they launch, resulting in a critical performance gap between laboratory findings and reality. Therefore, we move away from the independent and identically distributed assumption (i.i.d.) and focus on reducing generalization errors by studying the inter-dependencies among samples.

In this paper, we ask three questions (RQs) and propose our answers. The first question (RQ 1) is: “does higher diversity among training samples lead to better generalized NLP model learning?”. Semantic diversity is a desirable aspect in human annotation, where training data sets should be large and diverse enough to learn the many ways the objects differ (Shankar et al., 2017; Wu et al., 2018; Merler et al., 2019; Schumann et al., 2021), like intrinsic facial diversity in face recognition data, since every face is different. We show that diversity ensures that the training data can provide sufficiently discriminative information for the model (Gong et al., 2019), and thus give a more accurate prediction. Intuitively, for a fixed number of samples, the more semantic meaning they cover, the more information they contain, and thus the more effective they are for learning. We will use two examples to elaborate on our rationale.

Let us first take one explanatory example of $V = \{\text{be, cheerful, happy, stay}\}$, a four single-word dataset that forms two semantically close clusters: $A = \{\text{be, stay}\}$ and $B = \{\text{happy, cheerful}\}$. Our hypothetical NLP task is to generate sentences from $V$. From $\binom{4}{2}$ word combinations, four sentences are meaningful: “stay happy”, “stay cheerful”, “be happy”, “be cheerful”. Our dataset selection task is to find two single-word samples from $V$ that generate the most sentences. If we select both single-word samples from $A$, which are semanti-
We equipartitioned 42K training sentences into five subsets and compute their diversity using the sum of pairwise Cosine distances of sentence embeddings (i.e., our first method, see next paragraph and Section 2.2.1). We sort the subsets according to their diversity scores. The first subset has the lowest diversity ($3.18e^5$), the second subset has the second-lowest diversity ($3.26e^7$), etc. We then train five language models on each subset 1, and computes the perplexity (PPL) and out-of-vocabulary (OOV) on ten randomly sampled subsets of the test set. PPL and OOV are typical indicators of how training data is generalized (Rastogi et al., 2020; Müller et al., 2019). Our results show that the diversity of the subset strongly negatively correlates to both PPL and OOV rate, where the subset with the highest diversity is more likely to generalize on a random test and has fewer unseen words.

With the preliminary evidence showing that a training set with higher diversity leads to more generalized learning, we propose our second question (RQ 2): “how to measure the diversity of a sample set?” We introduce three methods to quantify training dataset diversity. The first method is based on the notion of dispersion, also known as Max-Sum Diversification (Cevallos et al., 2016). By viewing each training sample as a data point in the sentence embedding space, we maximize the dispersion of the training set, which is the sum of pairwise Cosine distances of sentence embeddings, as in Figure 2 (2). The larger the sum, the more informative the subset is.

The pairwise geometric distance method has a worst-case complexity of $O(n^2)$, and to accelerate the computation, we introduce our second method (Figure 2 (3)) that measures diversity using the volume of a convex hull around the samples, namely the volume that encloses all vertices, taking each sentence embedding as a vertex. Its quickhull (Barber et al., 1996) implementation has an average-case complexity of $O(n \log n)$ and a worst-case complexity of $O(n^2)$.

The above two methods compute diversity based on distances in the metric space. We can obtain global information content by measuring the uncertainty of the semantic distances between samples based on frequencies. Our third method combines our view of geometric distance and entropy in NLP. As depicted in Figure 2 (4), we apply Graph entropy, $H(G, P)$, an information-theoretic function on graph $G$ with a probability distribution $P$ on its vertex set (Dehmer and Mowshowitz, 2011; Rezaei, 2013). Here, each sentence in the subset is a node in the graph, and the Cosine distance of the two-sentence embeddings is the weighted edge between the two nodes. Empirically, all these three diversity measures consistently outperform the baselines. The Max Dispersion (MD) and Graph Entropy (GE) are more robust in various domains with the best performance overall because they account for every pair of distances in the training set.

After we have defined our diversity measure, we ask our third question (RQ 3): “how can we optimize the subset to maximize the information content?” Subset selection is NP-hard in general (Qian et al., 2016; Davis et al., 1997). We conjecture that this holds true for every objective function, including the notion of diversity introduced in this paper, and is the reason why we use actor-critic reinforcement learning (Konda and Tsitsiklis, 2000). We equipartition the training data into mini-batches.

---

1The language model is trained on Transformer with 2 heads, 2 hidden layers, 200 embedding size, and 200 hidden units on each subset for 10 epochs. Each test set subset contains 1000 sentences.
and simultaneously learn a policy network (as an actor) to select data and a value network (as a critic) to estimate future returns using the diversity measure as an evaluation reward. Our actor-critic data selection method has the advantages of low variance updates as well as credit assignment and significantly outperforms domain adaptation (Liu et al., 2019) on various domains without requiring any target domain knowledge. The architecture is shown in Figure 2 and Section 2.

In summary, our work mainly contributes to the following: (1) Modeling sample dependency for effective data sampling using diversity; (2) Measure dataset diversity using max dispersion, convex hull volume, and graph entropy; (3) Introduction of the Advantage Actor-Critic reinforcement learning framework to select informative samples and enhance NLP accuracy and generalization.

The rest of the paper is organized as follows. In Section 2.1, we detail Diversity Advanced Actor-Critic reinforcement learning (DAAC). Then in Section 2.2, we introduce different diversity measures. Afterward, in Section 3, we empirically verify the generalization and accuracy improvement using DAAC. We discuss related work in Section 4 and conclude the paper in the last section.

## 2 Diversity Advanced Actor-Critic

We now present the details of our Diversity Advanced Actor-Critic (DAAC) algorithms. The details of data selection and fine-tuning process are depicted in Figure 2. Our task model $F$ (any Deep Learning-based NLP model, such as Transformer, etc.) is pre-trained on the full training data set $X = \{x_i\}_{i=1}^n$, where $x_i$ is a sentence, $n$ is training set size. Then, as in Liu et al. (2019), we shuffle and randomly partition $X$ into $T$ disjoint data batches so that $X = \{B_t\}^T_{t=1} = \{B_1, B_2, ..., B_T\}$, with $B_t = \{x_{(t-1)n+1}, x_{(t-1)n+2}, ..., x_{tn}\}$. $n$ is the integer division of $n$ by $T$, and $T \leq t$. If $\mod(n, T) \neq 0$, then the last batch has a variable size of $\mod(n, T)$ and collects the remaining sentences. For each batch, we select and denote $\hat{B}_t = \{x_i\}_{i=1}^o | x_i \in B_t\}$ as the selected data with size $o$. After obtaining $\hat{B}_t$, we use $\hat{B}_t$ to fine-tune $F$. $F$ and its encoder $g$ is updated on $\hat{B}_t$ for $T$ times in an epoch, and each update is based on the previous checkpoint.

Our goal is to improve our task model $F$ on any
test domain whose distribution is different from
the source training set $X$, by learning an effective
subset of $X$ to fine-tune $F$. Figure 2-(1) shows our
new model.

Our framework is based on Actor-Critic rein-
forcement learning. Our Actor-Critic method has
the following properties, which are very desirable
to achieve our goals: variance deduction, efficient
sampling, and credit assignment (Konda and Tsit-
siklis, 2000; Grondman et al., 2012). Our frame-
work consists of policy and value networks jointly
and dynamically learned together with the task
model using the advantage error computed from
the reward function. In the following context, we
will first introduce our Actor-Critic algorithm (Sec-
tion 2.1) and then our three different reward func-
tions based on diversity (Section 2.2).

2.1 Advantage Actor Critic

2.1.1 Markov Decision Process

We cast the data selection as a reinforcement
learning problem with a Markov Decision Process
(MDP). Our data selection policy $\pi$, a mapping
from states to actions, serves as an agent to interact
with an environment that constitutes an NLP model,
we call task model $F$ over $T$ time steps. At each
time step $t$, a state $s_t$, an action $a_t$ and a reward $r_t$
are collected.

First, the encoder $g$ inside the NLP model (e.g.
an embedding layer in LSTM, or an encoder in
transformer) transforms a batch of data $B_t$ into its
embedding $s_t$ ($s_t = g(B_t)$ in Fig 2, e.g. vector
representations of the encoder output of a sentence).

Secondly, the policy $\pi$ outputs a probability distri-
bution for the batch of state $s_t$, so that each sen-
tence is associated with a probability representing
how likely it is going to be selected. The selected
subset, denoted as $B_t$, is then obtained by Bernoulli
sampling each sentence in the state $s_t$. For example,
for the $k$-th sentence in the batch with probability
$p_k$, we generate a random number between 0 and 1.
If $p_k$ is larger than the random number, then the $k$-
th sentence is selected; otherwise, it is not selected.
The result of Bernoulli sampling is represented as
a vector $a_t$, where each value in it is either 0 or 1
representing each sentence in the batch not being
or being selected.

Thirdly, task model $F$ as well as encoder $g$ are
finetuned by the selected subset $B_t$. In the mean-
time, a scalar reward $r_t = R(g(B_t))$ is calculated
by designed diversity reward functions $R$ which
we give definitions in Section 2.2.

Finally, the policy agent $\pi$ updates its weights
using the collected $s_t$, $a_t$ and $r_t$, where state $s_t$
is the encoded representation of a batch of data $B_t$,
action $a_t$ is a vector with each value of either 0
or 1 representing each sentence in the batch not being
or being selected, and the scalar reward $r_t$ measures
the diversity of selected batch $B_t$. This
optimization process is expanded in next section
2.1.2.

2.1.2 Training algorithm

We employ the Advantage Actor Critic algorithm
that uses $A(s_t, a_t)$, the advantage of action $a_t$
in state $s_t$ to scale the policy gradient. Specifically,
the advantage of action $a_t$ in state $s_t$ is defined in
Mnih et al. (2016) as

$$A(s_t, a_t) = Q(s_t, a_t) − V(s_t) \quad (1)$$

$$\approx \sum_{j=0}^{T−t} \gamma^{j} r_{t+j} − V(s_t) \quad (2)$$

where $\gamma \in (0, 1]$ is the discounting factor, and
we set its value as 0.99. $V$ is the value function
(critic) implemented as a value network in Figure 2-
(1).

The data selection policy $\pi$ (actor) is imple-
dented as a policy network, whose training object-
ive is

$$\nabla_\theta J(\theta) = E_{s \sim \pi} \nabla_\theta \log \pi_\theta(a_t|s_t; \theta) A(s_t, a_t; \theta, \theta_v).$$

The parameters of the policy network $\theta$ are updated by:

$$\theta_{v(t+1)} = \theta_v + \alpha \nabla_{\theta_v} V_\theta_v(r_t − V(s_t; \theta_v))^2$$

The parameters of value function $\theta_v$ is updated by:

$$\theta_v(t+1) = \theta_v + \alpha \nabla_{\theta_v} (r_t − V(s_t; \theta_v))^2 \quad (4)$$

The policy network $\pi(a_t|s_t; \theta)$ is a two-layer
network that has two nodes in the first layer and one
hidden output produced by the $\tanh$ activation
function and one softmax output. The value
network is a two-layer network that has eight nodes
Algorithm 1 DAAC Training Algorithm

Input: Epoch $L$, learning rate $\alpha$, discount factor $\gamma$, training set $X$, pre-trained task model $F$ (including encoder $g$), reward function $R$ (discussed in section 2.2)

Output: selected data, fine-tuned $F$, policy $\pi_\theta$, data value estimator $V_\theta$

1. Initialize data selection policy $\pi_\theta$ and value estimator $V_\theta$
2. for episode $l = 1$ to $L$
3. Shuffle (uniformly at random) all training samples;
4. Equipartition $X$ into $T$ (disjoint) sets with same size $n|T|: X = \{B_t\}_{t=1}^T = \{B_1, B_2, \ldots, B_T\}$;
5. Initialize an empty list: episode history $\Upsilon$
6. for all $B_t \in X$ (uniform transition probability) do
7. $s_t = g_t(B_t)$;
8. Obtain batch action $a_t$ by sampling based on $\pi_\theta(s_t)$;
9. $\hat{B}_t = \{(x_i)_{i=1}^o | a_i = 1\}$, where $o$ is selected sample size;
10. Update task model $F(g_t)$ by fine-tuning on $\hat{B}_t$;
11. $r_t = R(\hat{B}_t, g)$;
12. Store $(s_t, a_t, r_t)$ to episode history $\Upsilon$;
13. end for
14. for all $(s_t, a_t, r_t) \in \Upsilon$ do
15. Obtain $A(s_t, a_t)$ for each batch (Eq. 2);
16. Update policy weights $\theta$ (Eq. 3);
17. Update value estimator weights $\theta_v$ (Eq. 4);
18. end for
19. Clear episode history $\Upsilon$;
20. end for
21. return $F, \pi_\theta$ and $V_\theta$

2.2 Diversity Measures

The reward function in Section 2.1 is the diversity of the selected batch subset. We do not need any target domain knowledge, and the diversity is measured with the following three methods.

2.2.1 Max Dispersion

The dispersion of a set is the sum of all pair-wise distances within the set (Cevallos et al., 2016). Intuitively, maximizing the dispersion (denote as MD) of a set can enlarge the semantic coverage of the set, and thus diversify the content of the set. Formally, $x_i$ and $x_j$ are any two training sentences in a training set $G$, and their sentence embeddings are $g(x_i)$ and $g(x_j)$, respectively, we define the dispersion as

$$D(G) = \sum_{(x_i, x_j) \in G} d(x_i, x_j),$$

where $d(x_i, x_j) = 1 - \frac{\|g(x_i) - g(x_j)\|}{\|g(x_i)\||\|g(x_j)\|}$. The diversity of a set of training samples (sentences) is computed as the sum of each pairwise sample distance, which can be measured by any distance metrics, i.e., Cosine distance. We take the NLP task model encoder output as the sentence embedding to compute pairwise sample distance. The sentence embedding preserves pairwise sentence distance from the semantic space to the high-dimensional vector space. More details of the algorithm are shown in Algorithm 2, with a worst time complexity of $O(n^2)$, where $n$ is the number of samples in the set.

2.2.2 Convex Hull Volume

Instead of looking at each pair of sentences, we can simplify the computing process by looking at the boundary of a set of training samples in the high dimensional space and compute the volume of such space using Convex Hull Volume (denote as CV). Specifically, we treat each sentence as a point in the embedding space and approximate the volume of a convex body. In a Euclidean plane, given a finite set of points $Q$, it is sometimes interesting to determine its convex hull, namely the minimum convex polygon so that any point of $Q$ is either inside this polygon or at its border. There are a number of algorithms to compute the convex hull. Since our embedding is in high dimension, i.e., 256, we consider N-dimensional quickhull, which was introduced in Barber et al. (1996). Just like the quicksort algorithm, it has the expected time complexity of $O(n \log n)$ by divide and conquer approach, but may degenerate to $O(n^2)$ in the worst case. Details shown in appendix Algorithm 3.

2.2.3 Graph Entropy

Applying entropy on the distances, we introduce Graph Entropy (GE), following the so-called local measure of distance-based substructure entropy, details in Dehmer and Mowshowitz (2011). As

in the first layer and one hidden output produced by the ReLU activation function, and one linear output for the value network $V(s_t; \theta_v)$ following Mnih et al. (2016). The training algorithm is shown in Algorithm 1.
shown in Figure 2, graph entropy measures the uncertainty of the pairwise distance of sentence (or word) embeddings in the geometric space. To compute graph entropy of a set of training samples, we cast the sentence embeddings as vertices and distances between sentence embeddings as edges to form a clique. The distance entropy $I$ of each sample $x$ is defined as

$$ I(x) = -\sum_{i=1}^{M} f(d(x, x_i)) \log(f(d(x, x_i))). $$

$M$ is the number of sentences in the set, $d(x, x')$ is the Cosine distance between the encoder output vector representation of $x$ and $x'$, $f(d(x, x')) = \frac{d(x, x')}{\sum_{i=1}^{M}d(x, x_i)}$ is the relative distance of edge $d(x, x')$ (instead of words). The graph entropy of entire set is then defined as the sum of all distance entropy $I$:

$$ G(G) = \sum_{i=1}^{M} I(x_i). $$

For a set with $n$ data samples, there are $\binom{n}{2}$ edges in the resulting graph. The time complexity of graph entropy is thus $O(n^2)$ by looping through each edge in the graph. Details are shown in Algorithm 5.

3 Experiments

We describe our experimental details and demonstrate that our DAAC improves baselines in three NLP applications, including two recognition tasks: sentiment analysis, named entity recognition (NER), and one generation task of language modeling, without the knowledge of the target domain. We give hyperparameters of DAAC in appendix.

Finetuning vs. Data Selection We introduce our four models based on DAAC. The first three models finetune a pretrained NLP model using max dispersion MD, convex hull volume CV and graph entropy GE as reward function. As a comparison, we add the fourth model trained on selected data from scratch that does not rely on a pretrained model, denoted as data selection DS. Specifically, we train A2C using MD as reward function for 150 epochs first, and use the trained A2C to select data samples from all source data, then use only the selected data to train the NLP model from scratch. We select 3833 (63.8%) and 7630 (54.3%) samples from the source domain of sentiment analysis and NER respectively, and 21017 (49.9%) samples from Penn Treebank and 18043 (49.1%) samples from WikiText-2 in language modeling experiment.

Baselines We compare our models with five baselines: 1) All The models are trained on all source data; 2) Rand The models are trained on randomly selected 50% source data; 3) Marginal transfer learning (Mtl) by Blanchard et al. (2021), a domain generalization framework using kernel methods to augment feature space. 4) MetaReg (Meta) by Balaji et al. (2018), a domain generalization method using meta-learning. 5) Sim by Liu et al. (2019) that uses similarity features between target test domain and source training domains. To be noted, Sim is a domain adaptation method, which requires the usage of target domain data, while other baselines and our four models do not use target domain data.

3.1 Sentiment Analysis

Settings We use the Amazon product review dataset (Blitzer et al., 2007) for the sentiment analysis task. The sentiment analysis baseline is a CNN classifier (Kim, 2014). We pretrain the CNN for two epochs as (Liu et al., 2019) for a fair comparison.

Results Table 1 shows the results on amazon domains. Max dispersion outperforms other diversity measures in 11 out of all 22 unseen domains, and the Data Selection achieves a most boost of 38.0% (relative) and 23.8% (absolute) percent in the tools domain. This observation aligns with previous findings that 60%-70% important samples can perform similarly or better than training on the entire dataset.
3.2 Named Entity Recognition

**Settings** We use the CoNLL2003 English NER dataset (Sang and Meulder, 2003) as source training set and the five domains from CrossNER dataset (Liu et al., 2020) as test sets, which has specialized entity categories for each domain. We finetune the pretrained BERT model (Devlin et al., 2018) on source training set by adding a linear layer on top of the hidden-states output and then evaluate the F1-scores on five test domains.

**Results** The result is reported in Table 2. Graph Entropy achieves best performance among all the three diversity measures. In particular, Data Selection performs better than training with all data directly (All) and random selection (Rand), which means important samples are selected and noisy samples are filtered out.

3.3 Language Modeling

**Settings** Our baseline is a Transformer language model (Vaswani et al., 2017) with default hyperparameters. We experiment with two moderate size datasets WikiText-2 (Merity et al., 2016) and Penn Treebank. As for evaluation, we report perplexity scores on two translation datasets from different domains, IWSLT’17 (TED talk english) and WMT Biomedical’21 (english). The baseline models are trained using the fairseq toolkit (Ott et al., 2019) and stop training until the validation perplexity score does not improve for 5 epochs.

**Results** The evaluation results are shown in Table 3. The perplexity on two test domains have been improved. Specifically, test perplexity of Bio’21 has an improvement of 44.65 (37.9% relative improvement) while trained by our selected data. The comparison with randomly selected result (Rand) further proves the effectiveness of data selected by DAAC.

3.4 Ablation Analysis

**MD vs. GE** In our three experiments we observe Max Dispersion (MD) performs best in sentiment analysis and language modeling, and Graph Entropy measure (GE) achieves the best performance in named entity recognition. We infer the smaller improvement of convex hull volume might be due to error propagation of the quickhull algorithm.

Regarding the reason why GE performs best in named entity recognition, we first analyze the difference of operations on sentence embeddings for GE and MD. Since GE is calculated by the sum of distance entropy of each sentence (sample), there are more operations (e.g., log and normalization) depending on the sentence embedding input than MD. Thus, performance of GE is more relying on the precision of sentence embeddings compared to that of MD. Then looking into the key difference between the task of NER and other tasks, the source of sentence embeddings are apparently different. In NER, sentence embeddings are trained by BERT, while in sentiment analysis they are trained by CNN. As a result, we conjecture the more precise sentence embeddings trained by BERT lead to the better performance of GE on NER task. To verify this assumption, we quantitatively evaluate the quality of sentence embeddings generated by finetuned CNN and finetuned BERT on semantic textual similarity (STS) benchmarks STS-B-Dev (Cer et al., 2017), STS14 and STS15 (Agirre et al., 2014, 2015). We report the Spearman’s rank correlation between the cosine similarity of sentence embeddings using the SentEval toolkit (Conneau and Kiela, 2018). The results in table 4 show preliminary evidence that our conjecture holds. In short, MD can be consi-
Figure 3: Training/Validation accuracy change over epochs. Validate set is gourmet food domain of unprocessed Amazon product reviews. Training set is a joint set of dvd, kitchen and books, and target domain in Sim is electronics. MD can help alleviate overfitting on test domains compared to the domain adaptation method Sim (Liu et al., 2019)

<table>
<thead>
<tr>
<th></th>
<th>STS-B-Dev</th>
<th>STS14</th>
<th>STS15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetuned-CNN</td>
<td>29.92</td>
<td>23.26</td>
<td>19.77</td>
</tr>
<tr>
<td>Finetuned-BERT</td>
<td>32.02</td>
<td>24.54</td>
<td>23.06</td>
</tr>
</tbody>
</table>

Table 4: Spearman’s rank correlation between the cosine similarity of sentence embeddings generated by finetuned-CNN and finetuned-BERT. Numbers are reported as $\rho \times 100$.

Overfitting  We plot the validation accuracy curve of the sentiment classifier on one test domain, as shown in Fig 3. The validation accuracy of Sim (Liu et al., 2019) model degrades faster and more significant than DAAC. The gap of blue and red area, which is the gap between training accuracy and validation accuracy of models finetuned by Sim and MD, suggests DAAC can help alleviate overfitting on test domains compared to the domain adaptation method Sim.

Time  As discussed in 2.2, MD and GE both take account for the internal structure of a set of training samples and take time $O(n^2)$, where $n$ is size of the set. CV can reduce time to $O(n \log n)$ by utilizing divide and conquer, while accompanying with the degradation of performance. In theory, the entire training time of DAAC-finetuning is approximately $T$ times compared to training with all data, where $T$ is the time steps in Markov Decision Process, or the number of batches. In the finetuning setting, we follow Liu et al. (2019) and Yoon et al. (2019), using $T \in [2, 4]$ and finetune NLP model for 200 time steps. However, in the data selection (DS) setting, the training time is roughly 20 times more compared to the finetuning setting, since we train A2C for extra epochs (10000 time steps with $T = 60$, batch size 100) to ensure both the value and policy network converge. In practice, training with all source data directly (All) in sentiment analysis takes 131 seconds while finetuning with MD takes 217 seconds, and selecting data out then training (DS) takes 4774 seconds on one Tesla V100 GPU.

Visualization  Figure 4 visualizes data samples selected by our DAAC (diversified, red) and those randomly selected (nondiversified, blue), as well as the test data samples (unseen domains) in the embedding space. We observe that our DAAC selected samples are more widely spread, thus covering more semantic meaning for most test sets. Figure 4 (right) shows that DAAC selected samples has a larger convex hull volume than random selected samples after removing outliers. Furthermore, table 5 shows that our DAAC has a smaller size of out-of-vocabulary than the baseline on the test domains. Data samples are selected from the magazines domain of Amazon product reviews (Blitzer et al., 2007). We use max dispersion (MD) diversity measure, sentence-transformer toolkit (Reimers and Gurevych, 2019), and plot the embedding into two-dimension t-SNE.
4 Related work

Existing domain generalization methods related to data focus on data augmentation (Zhou et al., 2020; Qiao et al., 2020) and data generation (Wang et al., 2021). These methods increase the quantity of training set while our method utilizes existing data maximally by modeling the intrinsic sample dependency.

There have been several influential works (Moore and Lewis, 2010; Axelrod et al., 2011; Ruder and Plank, 2017a) on data selection that significantly contributed to today’s NLP state-of-the-arts. Fan et al. (2017) proposes a data filter based on deep reinforcement learning on image and sentiment classification task. Feng et al. (2018) implemented an instance selector using reinforcement learning to filter noisy data to improve the accuracy on natural language inference (Qu et al., 2019), sentiment analysis, part-of-speech tagging and dependency parsing (Liu et al., 2019). These work select the training data close to a given target domain for domain adaptation. In contrast, we aim to enhance the model generalization and increase the accuracy on any arbitrary domain.

As for diversity measures, Ruder and Plank (2017b) examines the effects of Shannon entropy in the transfer learning setting, but they do not consider content semantic meaning. Shi et al. (2021) uses determinantal point processes to select diverse data to reduce the labor of annotating training examples for dependency parsing, but the proposed diversity measure cannot be generalized to other NLP tasks. Other works incorporate the notion of diversity into topics like language representation (Chubarian et al., 2021), and ensemble language modeling (Duan et al., 2021).

5 Conclusion

We introduce Actor-Critic reinforcement learning rewarded with diversity measures to select effective training data that significantly enhances the accuracy of sentiment analysis, named entity recognition, and language modeling tasks across various domains. Without any target domain knowledge, our method outperforms the CNN, Transformer and BERT baselines. Our experiments show that modeling sample dependency by increasing training set diversity enhances the learning generalization and prediction accuracy.

Acknowledgments

We appreciate the National Science Foundation (NSF) Award No. 1747728 and NSF CRAFT Award, Grant No. 22001 to fund this research. We are also thankful for the support of the Google Cloud Research Program.

References


A Appendix

A.1 Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>NLP model</td>
</tr>
<tr>
<td>g</td>
<td>encoder</td>
</tr>
<tr>
<td>X</td>
<td>Training data set</td>
</tr>
<tr>
<td>x_i</td>
<td>sentence</td>
</tr>
<tr>
<td></td>
<td>Training set size</td>
</tr>
<tr>
<td>T</td>
<td>number of batches; maximum training steps in an epoch</td>
</tr>
<tr>
<td>B_i</td>
<td>batch</td>
</tr>
<tr>
<td>s_i</td>
<td>selected batch</td>
</tr>
<tr>
<td>s_k</td>
<td>batch state</td>
</tr>
<tr>
<td>a_t</td>
<td>single sentence state</td>
</tr>
<tr>
<td>a_k</td>
<td>action on batch B_t at time step t</td>
</tr>
<tr>
<td></td>
<td>action on single sample s_k at time step t</td>
</tr>
<tr>
<td>π</td>
<td>policy</td>
</tr>
<tr>
<td>r_t</td>
<td>reward at time step t</td>
</tr>
<tr>
<td>Q(s, a)</td>
<td>total future reward from t to T</td>
</tr>
<tr>
<td>V(s)</td>
<td>action value</td>
</tr>
<tr>
<td></td>
<td>expected future return following π since time step t</td>
</tr>
<tr>
<td>( \theta )</td>
<td>baseline function</td>
</tr>
<tr>
<td>( \nabla \theta )</td>
<td>parameters of policy network</td>
</tr>
<tr>
<td>( \nabla Q(\theta) )</td>
<td>objective function of policy</td>
</tr>
<tr>
<td>( L )</td>
<td>epoch number</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>learning rate</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>discount factor</td>
</tr>
<tr>
<td>( \mathcal{F} )</td>
<td>pretrained task model (including encoder g)</td>
</tr>
<tr>
<td>( \mathcal{E} )</td>
<td>episode record, including ( s_t, a_t, r_t )</td>
</tr>
<tr>
<td>( G )</td>
<td>a set of samples</td>
</tr>
<tr>
<td>( d )</td>
<td>cosine distance of embeddings</td>
</tr>
<tr>
<td>( M )</td>
<td>number of sentences in a set G</td>
</tr>
<tr>
<td></td>
<td>sentence length</td>
</tr>
<tr>
<td>( h(\cdot, n) )</td>
<td>n-gram entropy</td>
</tr>
</tbody>
</table>

Table 6: Notation table

A.2 Hyperparameters of A2C

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>learning rate</td>
<td>( 7e^{-4} )</td>
</tr>
<tr>
<td>discount factor</td>
<td>0.99</td>
</tr>
<tr>
<td>entropy coefficient</td>
<td>0</td>
</tr>
<tr>
<td>value function coefficient</td>
<td>0.5</td>
</tr>
<tr>
<td>RMSProp epsilon</td>
<td>( 1e^{-5} )</td>
</tr>
<tr>
<td>number of steps (finetuning)</td>
<td>200</td>
</tr>
<tr>
<td>number of steps (data selection)</td>
<td>15000</td>
</tr>
</tbody>
</table>

Table 7: Hyperparameters of A2C

Algorithm 2 Max Dispersion set diversity [MD]

**Input:** A batch of training samples \( G = \{(s_i)_{i=1}^M\} \) of size \( M \).

**Output:** Dispersion of the batch \( D(G) \)

1. Initialize \( D(G) = 0 \).
2. Initialize an empty set \( S \).
3. for all \( s_i \in G \) do
4. 
5. Compute Cosine distance \( d(v(s_i), v(s_j)) \);
6. for all \( s_j \in G \) if \( s_j \neq s_i \) and tuple \((i, j)\) not in \( S \) and tuple \((j, i)\) not in \( S \) do
7. 
8. Normalize \( v(s_j) \) by softmax function;
9. Compute Cosine distance \( d(v(s_i), v(s_j)) \);
10. \( D(G) = D(G) + d(v(s_i), v(s_j)) \);
11. Add tuple \((i, j)\) to \( S \);
12. end for
13. end for
14. return \( D(G) \)

Algorithm 3 Quickhull algorithm

**Input:** A batch of training samples \( G = \{(s_i)_{i=1}^M\} \) with size \( M \); \( M \geq 2 \)

**Output:** Convex hull set \( h \)

1. Initialize empty set convex hull \( h \);
2. Initialize empty dictionary \( d \);
3. for all \( s \in G \) do
4. 
5. for all \( s' \in G \) that \( s' \neq s \) do
6. 
7. end for
8. end for
9. Sort \( d \) in descending order
10. Add the two sentence \( s_p \) and \( s_q \) that has the max distance to convex hull set \( h \). Line formed by \( s_p \) and \( s_q \) segment the space into left half \( S_1 \) and right half \( S_2 \);
11. Call subroutine FindHull(\( S_1, s_p, s_q \) )
12. Call subroutine FindHull(\( S_2, s_q, s_p \) )
13. return Convex hull set \( h \)

Algorithm 4 Subroutine: FindHull

**Input:** A batch of training samples \( S \); point \( p \) and \( q \)

1. From the given set of points in \( S \), find farthest point \( f \)
2. Add point \( f \) to convex hull set \( h \). Three points \( p, q, \) and \( f \) partition the remaining points of \( S \) into 3 subsets: \( S_0, S_1, \) and \( S_2 \) where \( S_0 \) are points inside triangle \( pqf \), \( S_1 \) are points on the right side of the line from \( p \) to \( f \), and \( S_2 \) are points on the right side of the line from \( f \) to \( q \).
3. Call FindHull(\( S_1, p, f \) )
4. Call FindHull(\( S_2, f, q \) )

A.3 Diversity algorithms

See 2, 3 and 5.

A.4 Sentiment analysis full results

See table 9 and 8.

Table 8: Sentiment Analysis Accuracy on one test domain[\%]. The “book” column is tested on the book domain, while using other three domains for training.

<table>
<thead>
<tr>
<th></th>
<th>books</th>
<th>dvd</th>
<th>electronics</th>
<th>kitchen</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o tgt</td>
<td>All</td>
<td>77.58</td>
<td>79.88</td>
<td>83.50</td>
</tr>
<tr>
<td></td>
<td>Rand</td>
<td>76.39</td>
<td>79.30</td>
<td>82.99</td>
</tr>
<tr>
<td></td>
<td>Mt</td>
<td>77.20</td>
<td>79.90</td>
<td>84.25</td>
</tr>
<tr>
<td></td>
<td>Meta</td>
<td>76.10</td>
<td>79.60</td>
<td>83.90</td>
</tr>
<tr>
<td>w/o tgt</td>
<td>MD</td>
<td>79.67</td>
<td>84.08</td>
<td>85.29</td>
</tr>
<tr>
<td></td>
<td>CV</td>
<td>79.47</td>
<td>83.08</td>
<td>84.58</td>
</tr>
<tr>
<td></td>
<td>GE</td>
<td><strong>80.08</strong></td>
<td><strong>85.58</strong></td>
<td><strong>87.88</strong></td>
</tr>
<tr>
<td>w/ tgt</td>
<td>Sim</td>
<td>78.97</td>
<td>82.07</td>
<td>82.82</td>
</tr>
</tbody>
</table>

Table 9: A2C Performance comparison
Algorithm 5 Graph entropy based set diversity [GE]

Input: A batch of training samples $G = \{ (s_i)_{i=1}^M \}$ with size $M$
Output: Graph entropy $\hat{G}(G)$
1: Initialize $\hat{G}(G) = 0$
2: Initialize empty dictionary $d$
3: for all $s \in G$ do
4: for all $s' \in G$ that $s' \neq s$ do
5: Compute Cosine similarity $d[|v(s) - v(s')|]$ between sentence embedding of $s$ and $s'$;
6: $d[s] = d[s] + d(v(s), v(s'))$
7: end for
8: end for
9: for all $s \in G$ do
10: Initialize distance entropy of $s$: $I(s) = 0$
11: for all $s' \in G$ that $s' \neq s$ do
12: Compute Cosine similarity $d(v(s), v(s'))$ between sentence embedding of $s$ and $s'$;
13: Compute relative frequency of distance $f(d(v(s), v(s')))$
14: Update $I(s)$ according to equations in section 2.3.3
15: end for
16: $\hat{G}(G) = \hat{G}(G) + I(s)$
17: end for
18: return $\hat{G}(G)$

<table>
<thead>
<tr>
<th>Domain</th>
<th>All</th>
<th>Rand</th>
<th>Mtl</th>
<th>Meta</th>
<th>MD</th>
<th>CV</th>
<th>GE</th>
<th>DS</th>
</tr>
</thead>
<tbody>
<tr>
<td>apparel</td>
<td>49.43</td>
<td>50.47</td>
<td>47.65</td>
<td>49.92</td>
<td>50.53</td>
<td>50.48</td>
<td>50.78</td>
<td>49.95</td>
</tr>
<tr>
<td>auto</td>
<td>56.48</td>
<td>57.07</td>
<td>59.46</td>
<td>56.32</td>
<td>76.81</td>
<td>76.94</td>
<td>75.49</td>
<td>78.03</td>
</tr>
<tr>
<td>baby</td>
<td>50.42</td>
<td>51.09</td>
<td>50.80</td>
<td>50.06</td>
<td>50.86</td>
<td>52.95</td>
<td>53.12</td>
<td>52.81</td>
</tr>
<tr>
<td>beauty</td>
<td>54.03</td>
<td>54.93</td>
<td>55.89</td>
<td>52.26</td>
<td>65.43</td>
<td>65.43</td>
<td>65.03</td>
<td>65.97</td>
</tr>
<tr>
<td>camera</td>
<td>49.82</td>
<td>50.30</td>
<td>50.05</td>
<td>49.78</td>
<td>49.91</td>
<td>49.98</td>
<td>49.66</td>
<td>50.32</td>
</tr>
<tr>
<td>phones</td>
<td>52.79</td>
<td>53.77</td>
<td>53.93</td>
<td>52.51</td>
<td>62.13</td>
<td>61.81</td>
<td>61.18</td>
<td>61.98</td>
</tr>
<tr>
<td>computer</td>
<td>55.17</td>
<td>56.16</td>
<td>56.20</td>
<td>52.99</td>
<td>66.09</td>
<td>66.00</td>
<td>65.54</td>
<td>67.54</td>
</tr>
<tr>
<td>food</td>
<td>56.52</td>
<td>56.93</td>
<td>60.73</td>
<td>58.41</td>
<td>80.57</td>
<td>80.40</td>
<td>78.94</td>
<td>81.76</td>
</tr>
<tr>
<td>grocery</td>
<td>53.06</td>
<td>54.81</td>
<td>57.86</td>
<td>57.73</td>
<td>72.23</td>
<td>72.04</td>
<td>71.59</td>
<td>73.10</td>
</tr>
<tr>
<td>health</td>
<td>50.92</td>
<td>51.10</td>
<td>49.95</td>
<td>51.03</td>
<td>49.94</td>
<td>49.96</td>
<td>51.11</td>
<td>49.68</td>
</tr>
<tr>
<td>jewelry</td>
<td>56.63</td>
<td>57.92</td>
<td>58.98</td>
<td>58.15</td>
<td>75.77</td>
<td>75.56</td>
<td>74.84</td>
<td>76.81</td>
</tr>
<tr>
<td>magazines</td>
<td>50.78</td>
<td>50.56</td>
<td>50.42</td>
<td>50.03</td>
<td>51.15</td>
<td>51.23</td>
<td>50.94</td>
<td>50.97</td>
</tr>
<tr>
<td>music</td>
<td>50.05</td>
<td>50.27</td>
<td>50.08</td>
<td>49.60</td>
<td>50.08</td>
<td>50.20</td>
<td>50.20</td>
<td>50.03</td>
</tr>
<tr>
<td>instrs</td>
<td>59.15</td>
<td>59.53</td>
<td>61.88</td>
<td>56.41</td>
<td>82.92</td>
<td>82.82</td>
<td>81.40</td>
<td>84.45</td>
</tr>
<tr>
<td>office</td>
<td>56.84</td>
<td>59.28</td>
<td>61.87</td>
<td>58.14</td>
<td>82.92</td>
<td>82.23</td>
<td>80.58</td>
<td>83.73</td>
</tr>
<tr>
<td>outdoor</td>
<td>55.81</td>
<td>57.39</td>
<td>58.23</td>
<td>54.25</td>
<td>72.92</td>
<td>72.60</td>
<td>72.44</td>
<td>74.37</td>
</tr>
<tr>
<td>software</td>
<td>49.70</td>
<td>50.45</td>
<td>50.73</td>
<td>49.92</td>
<td>52.30</td>
<td>52.37</td>
<td>52.12</td>
<td>51.90</td>
</tr>
<tr>
<td>sports</td>
<td>51.08</td>
<td>51.13</td>
<td>49.97</td>
<td>49.98</td>
<td>50.66</td>
<td>50.68</td>
<td>50.70</td>
<td>50.25</td>
</tr>
<tr>
<td>tools</td>
<td>55.95</td>
<td>56.04</td>
<td>62.50</td>
<td>48.81</td>
<td>85.71</td>
<td>85.71</td>
<td>83.33</td>
<td>86.31</td>
</tr>
<tr>
<td>toys</td>
<td>50.07</td>
<td>50.82</td>
<td>50.20</td>
<td>50.35</td>
<td>51.03</td>
<td>51.02</td>
<td>51.38</td>
<td>50.17</td>
</tr>
<tr>
<td>video</td>
<td>51.65</td>
<td>50.72</td>
<td>50.08</td>
<td>50.68</td>
<td>50.50</td>
<td>50.70</td>
<td>50.85</td>
<td>50.20</td>
</tr>
</tbody>
</table>

Table 9: Sentiment analysis accuracy [%] on unknown domains.