Reinforced Curriculum Learning on Pre-trained Neural Machine Translation Models

Mingjun Zhao[1], **Haijiang Wu**[2], **Di Niu**[1], **Xiaoli Wang**[2]
[1]University of Alberta, Edmonton, AB, Canada
[2]Platform and Content Group, Tencent, Shenzhen, China
zhao2@ualberta.ca, harywu@tencent.com, dniu@ualberta.ca, evexlwang@tencent.com

Abstract

The competitive performance of neural machine translation (NMT) critically relies on large amounts of training data. However, acquiring high-quality translation pairs requires expert knowledge and is costly. Therefore, how to best utilize a given dataset of samples with diverse quality and characteristics becomes an important yet understudied question in NMT. Curriculum learning methods have been introduced to NMT to optimize a model's performance by prescribing the data input order, based on heuristics such as the assessment of noise and difficulty levels. However, existing methods require training from scratch, while in practice most NMT models are pre-trained on big data already. Moreover, as heuristics, they do not generalize well. In this paper, we aim to learn a curriculum for improving a pre-trained NMT model by re-selecting influential data samples from the original training set and formulate this task as a reinforcement learning problem. Specifically, we propose a data selection framework based on Deterministic Actor-Critic, in which a critic network predicts the expected change of model performance due to a certain sample, while an actor network learns to select the best sample out of a random batch of samples presented to it. Experiments on several translation datasets show that our method can further improve the performance of NMT when original batch training reaches its ceiling, without using additional new training data, and significantly outperforms several strong baseline methods.

1 INTRODUCTION

Curriculum learning, as pioneered by (Bengio et al. 2009), aims to improve the training of machine learning models by choosing what examples to present and in which order to present them to the learning algorithm. Curriculum learning was originally inspired by the learning experience of humans (Skinner 1958) (Peterson 2004) (Krueger and Dayan 2009)—humans tend to learn better and faster when they are first introduced to simpler concepts and exploit previously learned concepts and skills to ease the learning of new abstractions. This phenomenon is widely observed in, e.g., music and sports training, academic training and pet shaping.

Example 1	
zh	xianzai ta zheng kaolv zhe huijia.
zh-gloss	Now he is thinking about going home.
en	He is thinking about going home now.
Example 2	
zh	wo yao chi niupai.
zh-gloss	I want eat steak
en	I want a steak. Get me a coke.

Table 1: Examples with accurate/inaccurate translations.

Without surprise, curriculum learning is found most helpful in end-to-end neural network architectures (Bengio et al. 2009), since the performance that an artificial neural network can achieve critically depends on the quality of training data presented to it.

Neural Machine Translation (Kalchbrenner and Blunsom 2013) (Sutskever, Vinyals, and Le 2014) (NMT) translates text from a source language to a target language in an end-to-end fashion with a single neural network. It has not only achieved state-of-the-art machine translation results, but also eliminated hand-crafted features and rules that are otherwise required by statistical machine translation. The performance of NMT has been improved significantly in recent years, as the NMT architectures evolved from the initial RNN-based models (Sutskever, Vinyals, and Le 2014) to convolutional seq2seq models (Gehring et al. 2017) and further to Transformer models (Vaswani et al. 2017).

However, since obtaining accurately labeled training samples in machine translation is often time-consuming and requires expert knowledge, an important question in NMT is how to best utilize a limited number of available training samples, perhaps with different lengths, qualities, and noise levels. Recently, the application of curriculum learning is also studied for NMT. (Platanios et al. 2019) propose to feed data to an NMT model in an easy-to-difficult order and characterize the "difficulty" of a training example by the sentence length and the rarity of words that appear in it. Other than using the straightforward difficulty or complexity as a criterion for curriculum design, (Wang et al. 2018) propose a method to calculate the noise level of a training example

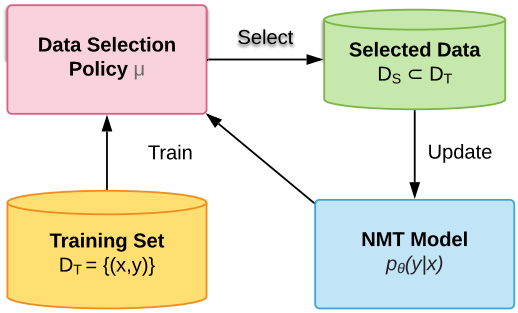

Figure 1: Illustration of the curriculum learning process. The RL policy μ is used to select samples D_S from the training set D_T. The selected data D_S is used to update a pre-trained NMT model $p_\theta(y|x)$.

with the help of an additional trusted clean dataset and train an NMT model in a noise-annealing curriculum.

A limitation of the existing curriculum learning methods for NMT is that they only address the batch selection issue in a "learn-from-scratch" scenario. Unfortunately, training an NMT model is a time-consuming task and sometimes could take up to several weeks (van der Wees, Bisazza, and Monz 2017), depending on the amount of data available. In most practical and commercial cases, a pre-trained model often already exists, while re-training it from scratch with a new ordering of batches is a waste of time and resources. In this paper, we study curriculum learning for NMT from a new perspective, that is given a pre-trained model and the dataset used to train it, to re-select a subset of useful samples from the existing dataset to further improve the model. Unlike the easy-to-difficult insights in traditional curriculum learning (Bengio et al. 2009), (Platanios et al. 2019), our idea is analogous to classroom training where a student first attends classes to learn general subjects with equal weights and then carefully reviews a subset of selected subjects to strengthen his/her weak aspects or to elevate ability in a field of interest.

Furthermore, while all the samples participate in batch training for the same number of epochs, it is unlikely that all data contribute equally to a best-performing model. Table 1 shows an example of two data samples from the dataset used in this paper, where Example 1 is accurately translated and can potentially improve the model better, while Example 2 is poorly translated (with unexpected words in target sentence) and may even cause performance degradation when fed to the model. The objective of our curriculum design is to identify examples from the existing dataset that may further contribute to model improvement and present them again to the NMT learning system. An overview of our proposed task is given in Figure 1, where useful data can be selected and fed to the system repeatedly to strengthen the model iteratively.

We formulate the data re-selection task as a reinforcement learning problem where the state is the features of b randomly sampled training examples, the action is choosing one of them, and the reward is the perplexity difference on a validation set after the pre-trained model is updated with the selected sample. Thus, the primary goal of the learning problem becomes searching for a data selection policy to maximize the reward. Reinforcement learning is known to be unstable or even to diverge when the action-value function is represented by a nonlinear function, e.g., a neural network. For the sake of alleviating instability, our proposed RL framework is built based on the Deterministic Actor-Critic algorithm (Silver et al. 2014). It consists of two networks, an actor network which learns a data selection policy, and a critic network which evaluates the action value of choosing each training sample while providing information to guide the training of the actor network. Besides introducing the framework, we also carefully design the state space by choosing a wide range of features to characterize each sample in multiple dimensions, including the sentence length, sentence-level log-likelihood, n-gram rarity, and POS and NER tagging.

Experiments on multiple translation datasets demonstrate that our method can achieve a significant performance boost, when normal batch learning cannot improve the model further, by only re-selecting influential samples from the original dataset. Furthermore, the proposed scheme outperforms a number of other curriculum learning baseline methods, including the denoising scheme based on the use of additional trusted data (Wang et al. 2018).

2 PROBLEM DEFINITION

In this section, we provide a brief background of NMT and formulate the curriculum learning task on pre-trained NMT models as a reinforcement learning problem.

Machine Translation can be considered a one-to-one mapping from a source sentence x to a target sentence y. In neural machine translation, a model parameterized by θ is searched for to maximize the conditional probability $p_\theta(y|x)$ over the training samples. Modern NMT models adopt an encoder-decoder architecture where an encoder encodes the source sentence x into a hidden representation h and a decoder predicts the next target word y_i given the hidden vector h and all previously predicted words $\{y_1, ..., y_{i-1}\}$. Thus the conditional probability is decomposed as

$$\log p_\theta(y|x) = \sum_{i=1}^{L} \log p_\theta(y_i|y_{<i}, h), \tag{1}$$

where L is the number of tokens in each target sentence. Given a training corpus D_T, the training objective of an NMT model is to minimize

$$J(\theta) = \sum_{(x,y)\in D_T} -\log p_\theta(y|x). \tag{2}$$

We consider curriculum learning on a pre-trained NMT model, where the goal is to improve an existing model $p_\theta(y|x)$ by selecting a subset D_S from the training dataset D_T that led to $p_\theta(y|x)$. As compared to training from scratch, we take advantage of both the versatility of normal batch learning in the initial pre-training stage and a carefully selected curriculum for targeted model improvements.

Specifically, our objective is to find an optimal policy μ_ϕ to select D_S from D_T and update $p_\theta(y|x)$ with D_S such that the performance of the updated model is maximized, i.e.,

$$\begin{aligned} \max_{\phi} \quad & \textit{perf}\Big(p_{\theta'}(y|x,\phi)\Big), \\ \text{s.t.} \quad & p_{\theta'}(y|x,\phi) = \textit{train}\Big(p_\theta(y|x), D_S(\phi)\Big), \\ & D_S(\phi) = \mu_\phi(D_T), \end{aligned} \tag{3}$$

where $D_S(\phi)$ is a subset selected from D_T using policy μ_ϕ, $p_{\theta'}(y|x,\phi)$ is an updated model after training $p_\theta(y|x)$ with $D_S(\phi)$, and *perf* indicates the performance of a model, e.g., measured by BLEU or Perplexity.

The main challenge is to identify and select the most beneficial data samples from D_T. A naive way is to evaluate the BLEU improvement on a validation set brought by every single data sample in the training set and select the ones that improve the BLEU the most. However, this method is extremely costly and is not scalable to large datasets.

To obtain a generalizable data selection policy, we formulate the task as a reinforcement learning problem in which the environment is composed of both the dataset D_T and the model $p_\theta(y|x)$. The RL agent aims to learn a policy μ_ϕ which decides which sample to select when presented with a random batch of samples. In our framework, a state s corresponds to the representation of both a data batch to select from and the NMT model, a refers to the action of selecting the best data sample from the batch, and r is the performance improvement of the NMT model on validation set after being updated with the selected sample.

The RL agent is trained through interacting with the environment by repeatedly performing the following: 1) receiving a state s containing a random batch of samples, 2) providing an action a back to the environment according to its trained policy, and 3) updating the policy using a feedback reward r given by the environment. Once the policy is trained, it can be used to select data from an arbitrarily large dataset and is scalable.

3 METHODS

In this section, we describe our Deterministic Actor-Critic framework for curriculum learning, as well as the design of the state, action, and reward in detail.

3.1 Model Overview

For the model design of the RL agent, we choose the Deterministic Actor-Critic algorithm.

Actor-critic (Konda and Tsitsiklis 2000) is a widely used method in reinforcement learning combining both an actor network μ_ϕ that outputs an action $a = \mu_\phi(s)$ given a state s to maximize the reward, and a critic network Q_w that predicts the action value $Q_w(s,a)$ of a state-action pair (s,a). The actor learns a near-optimal policy via policy gradient, while the critic estimates the action-value guiding the update direction of the actor. Compared with actor-only methods like REINFORCE (Williams 1992), the existence of the critic reduces the update variance and accelerates convergence.

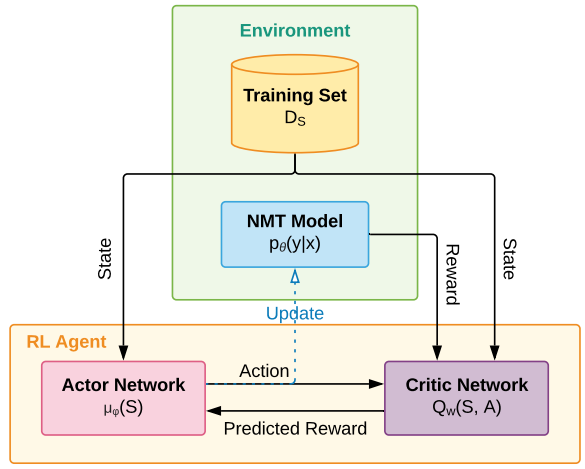

Figure 2: The proposed RL framework.

As our reward calculation involves evaluating the updated NMT model on a validation set and is thus expensive, we exploit a memory replay buffer to increase sample efficiency. Furthermore, we choose a deterministic policy setting as opposed to stochastic policy due to the fact that the deterministic policy gradient can be calculated much more efficiently as shown in (Silver et al. 2014).

The update of the framework is illustrated in Figure 2. The critic network Q_w takes a state-action pair (s,a), evaluates the action value, and outputs a predicted reward $\tilde{r} = Q_w(s,a)$, and updates the parameters supervised by the actual reward r from the environment. Note that the critic network provides an immediate reward per iteration. As a result, we do not need to employ additional techniques, e.g., Temporal-Difference (TD) (Tesauro 1992), to approximate the long-term reward. The objective of the critic network is thus to minimize the squared error of δ between the predicted reward and the actual reward, given by

$$\delta_t = r_t - \tilde{r}_t = r_t - Q_w(s_t, a_t). \tag{4}$$

The update of parameters w is achieved through gradient descent as follows:

$$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w Q_w(s_t, a_t). \tag{5}$$

where α_w corresponds to the learning rate of parameters w.

The actor network μ_ϕ takes in a state s from the environment, applies the learned policy, and outputs a corresponding action a. The objective of the actor network is to learn an optimal policy generating the proper action to maximize the predicted reward $Q_w(s,a)$. Policy gradient is used to update the parameters ϕ, i.e.,

$$\begin{aligned} \phi_{t+1} &= \phi_t + \alpha_\phi \nabla_\phi Q_w(s_t, a_t)|_{a=\mu_\phi(s)} \\ &= \phi_t + \alpha_\phi \nabla_\phi \mu_\phi(s_t) \nabla_a Q_w(s_t, a_t)|_{a=\mu_\phi(s)}. \end{aligned} \tag{6}$$

Algorithm 1 summarizes the overall learning process of the proposed framework. In each round of data selection

Algorithm 1: The Proposed Method

Input: Training set D_T and an NMT model $p_{\theta_0}(y|x)$
Output: A better performing NMT model $p_{\theta_K}(y|x)$

1 **for** $t = 0, \ldots, K-1$ **do**
2   **for** *number of RL training iterations* **do**
3   Sample b examples from D_T and form state s
4   Generate action $a = \mu_\phi(s)$
5   Compute predicted reward $\tilde{r} = Q_w(s, a)$
6   Update $p_{\theta_k}(y|x)$ with selected sample to get $p_{\theta'_k}(y|x)$
7   Calculate the perplexity difference on validation set between $p_{\theta_k}(y|x)$ and $p_{\theta'_k}(y|x)$ as reward r
8   Update Q_w using Eq. (5)
9   Update μ_ϕ using in Eq. (6)
10   Select data D_S from D_T using μ_ϕ
11   Update $p_{\theta_k}(y|x)$ with D_S to get $p_{\theta_{k+1}}(y|x)$

(with K rounds in total), we first train the RL agent for an adequate number of iterations. In each iteration, we derive a state, an action, and a reward in lines 4–7 and update the actor network and the critic network in line 8 and line 9, respectively. After the RL agent is fully trained, we apply the learned policy to select a subset D_S and use D_S to update the NMT model $p_{\theta_k}(y|x)$ and move to the next round. Usually one or two rounds are sufficient.

Figure 3 demonstrates the network structure of the RL agent. A *feature network* is shared between the Actor Network μ_ϕ and the Critic Network Q_w. It takes in the raw features of the sampled batch of b examples, where each feature of each sample goes through an independent single-layer MLP. The concatenation of the outputs constitutes the state representation s. Note that different examples in the sampled batch share the same network weights.

The Actor Network μ_ϕ is composed of a two-layer MLP and computes a score for each example in the sampled batch of b examples based on the input state s, and outputs the action a as a probability vector representing the probability of each example being selected, by taking a softmax operation over the computed scores of b examples.

The Critic Network Q_w also has two layers and calculates the action value of a given state-action pair (s, a), where the action a is the output of the Actor Network, i.e., $a = \mu_\phi(s)$, and is concatenated to the second layer of the critic network.

Note that although the feature network is shared between both the actor network μ_ϕ and the critic network Q_w, we only update it with the critic network to reduce training instability.

3.2 State

The state s is meant to be a full summarization of the environment including a data batch of b examples to select from and information about the pre-trained model. However, the number of parameters in the pre-trained model is too large to be included in the state directly at each time step. Thus, we need to find a representation that can represent both the

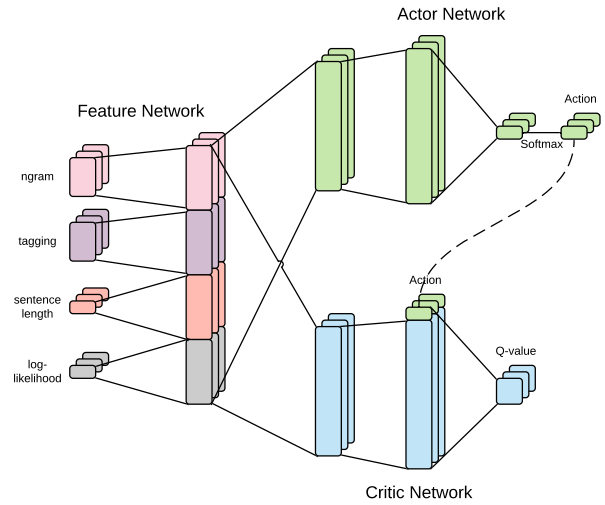

Figure 3: The network structure of the RL agent, where the depth represents a batch (of 3 samples in this illustration).

samples to be selected and the existing model using a limited number of parameters. In our state design, we focus on three different dimensions, namely *informativeness*, *uncertainty*, and *diversity*. We use the sentence length as a measure for informativeness, the sentence-level log-likelihood for uncertainty, and the n-gram rarity together with NER and POS taggings for diversity.

The most intuitive representation feature of a parallel sentence is the sentence length, i.e. the number of tokens in a sentence. This simple scalar roughly measures the amount of information involved in a sentence and is also used in (Platanios et al. 2019) as a "difficulty" estimate.

Following the intuition that examples yielding large uncertainty can benefit the model performance (Li and Sethi 2006), another feature we use is the sentence-level log-likelihood $L_{p_\theta(y|x)}$ calculated by the pre-trained model $p_\theta(y|x)$ by summing up the log probability of each word y_i of the target sentence:

$$L_{p_\theta(y|x)} = \log p_\theta(y|x) = \sum_{y_i \in Y} \log p_\theta(y_i|x). \qquad (7)$$

(Yang et al. 2015) and (Sener and Savarese 2018) suggest that selecting samples that are farthest away from the existing dataset can benefit model training. Incorporating this idea, we further utilize two other feature vectors, n-gram rarity and taggings, to represent the similarity between a given sample and the entire training set.

For n-gram rarity, we use $n = \{1, 2, 3, 4\}$. Given all the sentences in D_T, we define the relative frequency for a unique n-gram $g_j^{(n)}$ in D_T as

$$f^{(n)}(g_j^{(n)}) \triangleq \frac{1}{N^{(n)}} \sum_{s_k \in D_T} \sum_{g_i^{(n)} \in g_{(s_k)}^{(n)}} \mathbb{1}_{g_i^{(n)} = g_j^{(n)}}, \qquad (8)$$

where $j = 1, ..., \#\{\textit{unique n-grams}\}$, s_k is a sentence from D_T, $g^{(n)}_{(s_k)}$ is all n-grams in s_k, and $N^{(n)}$ is the total number of n-grams in D_T. For n-gram representation of a sentence s_k, we form all the n-gram frequencies into a vector:

$$F^{(n)}(s_k) = \{f(g^{(n)}_1), ..., f(g^{(n)}_L) | g^{(n)}_l \in g^{(n)}_{(s_k)}\}. \quad (9)$$

In this paper, we use $n = \{1, 2, 3, 4\}$ and calculate the n-gram vectors for both the source and target sentences.

For taggings, we use Named Entity Recognition (NER) and Parts of Speech (POS) and apply ideas similar to our n-gram design. As the tagging of a word is dependent of the sentence that the word lies in, a same word may be tagged differently in various sentences. To give an example in POS tagging, the word "Book" can either be tagged as "NOUN" or "VERB" depending on its meaning in the sentence. If most occurrences of the word "Book" are tagged as "NOUN" in the training set, the model may not be able to correctly learn the second meaning of it. Therefore, the model should be fed more with samples in which "Book" is tagged as "VERB". In order to reflect this phenomenon, we define a tagging value for a word w and its current tag t as

$$v(w, t) \triangleq \frac{1}{N_w} N_{tag(w)=t}, \quad (10)$$

where N_w is the number of times the word w has appeared in the training set and $N_{tag(w)=t}$ is the number of times the word w is tagged with tag t among all its occurrences. Similarly, for both NER and POS taggings, the tagging values of words form a vector representation of a sentence:

$$V(s_k) = \Big\{v\Big(w_1, t_1\Big), ..., v\Big(w_L, t_L\Big) | w_l \in s_k\Big\}. \quad (11)$$

3.3 Action and Reward

In the task of curriculum learning, an action represents the process of data selection. (Fang, Li, and Cohn 2017) assume a stream-based setting where data examples come one by one in a stream, and design their action as making a decision on whether or not a single incoming data example should be selected. We argue that our problem setting is actually *pool-based* instead of stream-based, where a pool of data exists for selection and deciding on the selection of each individual sample per step would be inefficient. In (Kumar et al. 2019), a dataset is split into several bins according to a noise measure of data samples, and the action determines from which bin the next batch of training data should be selected. However, this method highly depends on an effective heuristic criterion for bin formation and is thus hard to generalize.

Therefore, we propose our action design which samples a batch of b data examples from D_T, computes a score for each example according to the trained policy, and choose the one with the highest score. Correspondingly, in our design, we can easily control the size of D_S by varying the batch size b, i.e., $|D_S| = |D_T|/b$, since we choose one out of b samples for each batch.

For the choice of reward signal, we use the performance improvement of the NMT model evaluated on the validation set after it is updated with the selected sample. *Perplexity* is used as the performance metric instead of BLEU as it is shown to be more consistent and less noisy (So, Le, and Liang 2019). We assign a reward of 0 to unselected samples.

Table 2: Description of evaluation datasets.

Dataset	Train	Val	Test	Src-Len	Tgt-Len
MTD	$1,000,000$	$1,892$	$1,892$	19.55	21.06
CASICTB	$994,248$	$2,002$	$2,001$	22.61	23.95
CASIA2015	$1,049,988$	$2,002$	$2,001$	23.89	25.19
NEU	$1,999,946$	$2,002$	$2,001$	16.95	18.54

4 EXPERIMENTS

In this section, we will first describe the datasets used in our evaluation and provide the implementation details along with the performance results.

4.1 Datasets & Metrics

To compare our proposed method with other curriculum learning methods on NMT task, we conduct comprehensive empirical analysis on several zh-en translation datasets:

- **MTD** is an internal news translation dataset with 1M samples in the training set and 1,892 samples in both the validation set and the test set. The average length of source sentences is 19.55 and the average length of target sentences is 21.06.
- **CASICTB, CASIA2015, NEU** are three independent datasets with 1M, 2M, and 2M examples from different data sources in WMT18 which is a public translation dataset in news area with more than 20M samples. We only use a part of data from WMT18 to evaluate our method. All three datasets share the same validation set *newsdev2017* and the test set *newstest2017* both composed of 2k samples.

Table 2 summarizes the details of the experimental datasets. The columns titled "Train", "Val", and "Test" correspond to the number of examples in training, validation, and test sets. "Src-Len" and "Tgt-Len" correspond to the average sentence length of source language and target language respectively.

For evaluation metrics, we use Perplexity for the reward calculation in RL agent training, and report BLEU (Papineni et al. 2002) for the final performance of the NMT models.

4.2 Experimental Settings

We implement our models in PyTorch 1.1.0 (Paszke et al. 2017) and train the model with a single Tesla P40. We utilize NLTK (Bird, Klein, and Loper 2009) to perform POS and NER taggings.

For the NMT model, we use the OpenNMT implementation (Klein et al. 2017) of Transformer (Vaswani et al. 2017). It consists of a 6-layer encoder and decoder, with 8 attention heads, and 2,048 units for the feed-forward layers. The multi-head attention model dimension and the word embedding size are both set to 512. During training, we use Adam optimizer (Kingma and Ba 2015) with a learning rate of 2.0 decaying with a noam scheduler and a warm-up steps of 8,000. Each training batch contains 4,096 tokens and is

selected with bucketing (Kocmi and Bojar 2017). During inference, we employ beam search with a beam size of 5.

For the RL agent, we use an Deterministic Actor-Critic architecture and build our system based on (Shangtong 2018). In our framework, we use several tricks proven to be effective to RL training including a memory replay buffer of size 2,500, a warm-up phase of 500 steps, and a target network which is updated by mixing weights with the on-line network with a mix factor of 0.1. For calculating rewards, we train the NMT model with the single selected sample using SGD and a learning rate of 1e-4.

The feature network maps data features of sentence length, sentence-level log-likelihood, taggings, and n-gram rarity to vectors of size 1, 8, 16, and 32 respectively with a FC layer, and concatenates them together as a shared state representation. The actor network is composed of two FC layers with hidden sizes of 300 and 400. The critic network is designed the same as the actor network except that the output action from actor network is concatenated to the second layer. Relu is used as the activation function in each FC layer in this network.

In experiments, we conduct two rounds of RL agent training and data selection. In each round, the RL agent is trained for 20k steps and the best model with the highest sum of rewards during the last 1,000 steps is used for data selection. The RL agent keeps selecting data given randomly sampled batches from the training set D_T and feed the selected data to the NMT model until no performance improvement is observed. For the training process on selected data, we keep the NMT model's optimizer and learning rate setting unchanged.

We use different batch sizes of the sampled batch b for the two rounds of training and selection with $b_1 = 16$ and $b_2 = 128$ indicating in the first round we select 1 sample from 16 and in the second round, from 128 samples we select the best one. This is because we think in order to achieve improvement further on the basis of the first round, a stricter data selection criterion must be applied.

4.3 Baselines

To make comparisons with other existing curriculum methods, we have conducted several baseline experiments.

We take the core ideas of existing curriculum learning methods of training on data samples with gradually increasing difficulty (Platanios et al. 2019) and gradually decreasing noise (Wang et al. 2018) and apply them to our setting with pre-trained models. We evaluate the following three baseline methods along with our proposed method.

- **Denoising** is a curriculum learning method of training an NMT model in a noise-annealing fashion (Wang et al. 2018). They propose to measure NMT data noise with the help of a trusted dataset which contains generally cleaner data compared to the training dataset. (Kumar et al. 2019) also utilize data noise in their curriculum design and achieve similar performance as (Wang et al. 2018). For the choice of the trusted dataset, we choose a subset of 500 sentences from the validation set newsdev2017 of CASICTB, CASIA2015 and NEU.
- **Sentence Length** is an intuitive difficulty measure used in (Platanios et al. 2019), since longer sentences tend to contain more information and more complicated sentence structure.
- **Word Rarity** is another metric for measuring the sample difficulty, as rare words appear less frequently in the training process and should be presented to the learning system more. The formula for calculating the word rarity of a sentence can be found in (Platanios et al. 2019).

For baseline experiments, the pre-trained NMT model is further trained on 20% of the original data, which are selected by the above criteria, i.e., the least noisy, the longest, and the highest word rarity, respectively.

4.4 Main Results

Table 3 compares the performance of our method with other baseline methods on different datasets evaluated using BLEU. The result shows that our proposed method significantly out-performs other baseline methods by a great margin. We conduct two rounds of training and update in our experiments. While the result of the first round surpasses almost all the baseline methods, our second round further improves the performance and achieves a final BLEU improvement of $+0.90$, $+0.60$, $+0.59$, and $+0.71$ on MTD, CASICTB, CASIA2015, and NEU respectively over the pre-trained model.

The reason of our success is due to our utilization of an RL framework to proactively select data samples that are potentially beneficial to the training of the NMT model. First, we formulate the task of curriculum learning on pre-trained NMT models as a reinforcement learning problem. Second, we construct an effective design of state, action and reward. Our state representation includes features of different dimensions of informativeness, uncertainty and diversity. Third, we propose a Deterministic Actor-Critic framework that learns a policy to select the best samples from the training set to improve the pre-trained model. By incorporating all these designs together, our proposed framework is able to achieve a significant performance enhancement on the pre-trained NMT model.

4.5 Analysis

We evaluate the impact of different modules and methods by ablation test on MTD dataset. Table 4 list the performance of our model variants with different features included.

We incrementally accommodate different features of examples to the state by first starting from sentence length and sentence-level log-likelihood as they are both scalars. The performance increased slightly by 0.26 BLEU compared with the pre-trained base model. Then we further accommodate n-gram rarity and POS and NER taggings to the state vectors, and observe a larger increase of performance of 0.42 and 0.60 respectively. Finally, we incorporate the second round of RL agent training and data selection on the basis of the result of first round, and achieve the best performance with a 0.90 BLEU increase. Note that a stricter selection policy is applied to the second round (128 choose 1) compared with the first round (16 choose 1).

Table 3: Performance comparison of our proposed method with other baseline methods using BLEU on different datasets.

Method	MTD	CASICTB	CASIA2015	NEU
Base	18.21 (+0.00)	13.43 (+0.00)	18.65 (+0.00)	20.06 (+0.00)
Sentence Length	18.34 (+0.13)	13.52 (+0.09)	18.75 (+0.10)	20.21 (+0.15)
Word Rarity	18.35 (+0.14)	13.49 (+0.06)	18.72 (+0.07)	20.17 (+0.11)
Denoising	18.42 (+0.21)	13.55 (+0.12)	18.86 (+0.21)	20.44 (+0.38)
Ours-1 Round	18.81 (+0.60)	13.60 (+0.17)	18.91 (+0.26)	20.38 (+0.32)
Ours-2 Rounds	**19.11** (+**0.90**)	**14.03** (+**0.60**)	**19.24** (+**0.59**)	**20.79** (+**0.73**)

Table 4: Ablation analysis on different features of our proposed framework on MTD.

Method	MTD
Base	18.21 (+0.00)
Senlen + Logp	18.47 (+0.26)
+ N-gram	18.63 (+0.42)
+ Tagging	18.81 (+0.60)
+ 2nd Round	**19.11** (+**0.90**)

5 RELATED WORK

High-quality machine translation corpus is costly and difficult to collect, thus it is necessary to make the best use of the corpus at hand. A straightforward method to achieve this goal is to remove the noisy samples in the training data, and train an new model with the clean ones. Unfortunately, it is hard to estimate the quality of a parallel sentence in the absence of golden reference (Fan et al. 2019). Moreover, (Wang et al. 2018) find that some of the noisy samples may yield some benefits to the model performance. Then they define a new method of computing noise level of a data example with the help of an extra trusted dataset and propose to train an NMT model in a noise-annealing curriculum.

Curriculum learning aims to organize the training process in a meaningful way by feeding certain samples to the model in certain training stage such that the model can learn better and faster (Bengio et al. 2009). They propose a simple strategy which organizes all training samples into bins of similar complexity and starts training from the easiest bin to include more complexed bins until all bins are covered. (Kocmi and Bojar 2017) apply this idea to NMT by binning according to simple features like sentence length and word frequency, and improve this strategy by restricting that each sample can only be trained once during an epoch. (Zhang et al. 2018) conduct empirical studies on several hand-crafted curriculum and adopt a probabilistic view of curriculum learning. (Platanios et al. 2019) further propose a competence function $c(t)$ with respect to training time step t as the indicator of learning progress and select samples based on both difficulty and competence. However, these heuristic-based approaches highly depend on hand-crafted curriculum and are hard to generalize.

Compared with heuristic-based approaches, RL-based policy learning models are trained end-to-end and do not rely on hand-crafted strategies. (Tsvetkov et al. 2016) use Bayesian optimization to learn a linear model for ranking examples in a work-embedding task. (Graves et al. 2017) explore bandit optimization for scheduling tasks in a multi-task problem.(Wu, Li, and Wang 2018) select examples in a co-trained classifier using RL. (Kumar et al. 2019) organize the dataset into bins based on the data noise proposed by (Wang et al. 2018) and utilize a DQN to learn a data selection policy deciding from which bin to select the next batch.

Different from the existing curriculum learning methods, our work focuses on learning a training curriculum with reinforcement learning for an existing pre-trained NMT model. We argue that existing curriculum learning methods are only applicable on train-from-scratch scenarios, and learning from an existing model can save training time.

Active learning (Settles 2009) is another related area which focuses on selectively obtaining labels for unlabeled data in order to improve the model with least labeling cost. (Haffari, Roy, and Sarkar 2009) (Bloodgood and Callison-Burch 2010) study active learning for Statistical Machine Translation using some hard-coded heuristics. (Fang, Li, and Cohn 2017) design an active learning algorithm based on a deep Q-network, in which the action corresponds to binary annotation decisions applied to a stream of data. (Liu, Buntine, and Haffari 2018) make use of imitation learning to train a data selection policy.

6 CONCLUSION

In this paper, we study curriculum learning for NMT from a new perspective, to re-select a subset of useful samples from the existing dataset to further improve a pre-trained model, and formulate this task as a reinforcement learning problem. Compared with existing curriculum methods only applicable on train-from-scratch scenarios, our setting saves training time by better utilizing the existing pre-trained models. Our proposed framework is built based on the deterministic actor-critic algorithm, and learns a policy to select examples that can improve the model the most. We conduct experiments on several zh-en translation datasets and compare our method with other baseline methods including the easy-to-difficult curriculum and the denoising scheme. Through rounds of training and data selection, our method achieves a significant performance boost on the pre-trained model, and out-performs all baselines methods by a great margin.

References

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, 41–48. ACM.

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Bloodgood, M., and Callison-Burch, C. 2010. Bucking the trend: Large-scale cost-focused active learning for statistical machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 854–864. Association for Computational Linguistics.

Fan, K.; Wang, J.; Li, B.; Zhou, F.; Chen, B.; and Si, L. 2019. "bilingual expert" can find translation errors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6367–6374.

Fang, M.; Li, Y.; and Cohn, T. 2017. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the 2017 EMNLP*, 595–605.

Gehring, J.; Auli, M.; Grangier, D.; and Dauphin, Y. 2017. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th ACL*, 123–135.

Graves, A.; Bellemare, M. G.; Menick, J.; Munos, R.; and Kavukcuoglu, K. 2017. Automated curriculum learning for neural networks. In *Proceedings of the 34th ICML-Volume 70*, 1311–1320. JMLR. org.

Haffari, G.; Roy, M.; and Sarkar, A. 2009. Active learning for statistical phrase-based machine translation. In *Proceedings of Human Language Technologies: NACCL 2009*, 415–423. Association for Computational Linguistics.

Kalchbrenner, N., and Blunsom, P. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1700–1709.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Klein, G.; Kim, Y.; Deng, Y.; Senellart, J.; and Rush, A. M. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Kocmi, T., and Bojar, O. 2017. Curriculum learning and minibatch bucketing in neural machine translation. In *Proceedings of RANLP 2017*, 379–386.

Konda, V. R., and Tsitsiklis, J. N. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*, 1008–1014.

Krueger, K. A., and Dayan, P. 2009. Flexible shaping: How learning in small steps helps. *Cognition* 110(3):380–394.

Kumar, G.; Foster, G.; Cherry, C.; and Krikun, M. 2019. Reinforcement learning based curriculum optimization for neural machine translation. In *Proceedings of the 2019 NACCL*, 2054–2061.

Li, M., and Sethi, I. K. 2006. Confidence-based active learning. *IEEE transactions on pattern analysis and machine intelligence* 28(8):1251–1261.

Liu, M.; Buntine, W.; and Haffari, G. 2018. Learning to actively learn neural machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, 334–344.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th ACL*, 311–318. Association for Computational Linguistics.

Paszke, A.; Gross, S.; Chintala, S.; and Chanan, G. 2017. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration* 6.

Peterson, G. B. 2004. A day of great illumination: Bf skinner's discovery of shaping. *Journal of the experimental analysis of behavior* 82(3):317–328.

Platanios, E. A.; Stretcu, O.; Neubig, G.; Poczos, B.; and Mitchell, T. 2019. Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 1162–1172.

Sener, O., and Savarese, S. 2018. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*.

Settles, B. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Shangtong, Z. 2018. Modularized implementation of deep rl algorithms in pytorch. https://github.com/ShangtongZhang/DeepRL.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 387–395.

Skinner, B. F. 1958. Reinforcement today. *American Psychologist* 13(3):94.

So, D.; Le, Q.; and Liang, C. 2019. The evolved transformer. In *International Conference on Machine Learning*, 5877–5886.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Tesauro, G. 1992. Practical issues in temporal difference learning. In *Advances in neural information processing systems*, 259–266.

Tsvetkov, Y.; Faruqui, M.; Ling, W.; MacWhinney, B.; and Dyer, C. 2016. Learning the curriculum with bayesian optimization for task-specific word representation learning. In *Proceedings of the 54th ACL (Volume 1: Long Papers)*, 130–139.

van der Wees, M.; Bisazza, A.; and Monz, C. 2017. Dynamic data selection for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 1400–1410.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, W.; Watanabe, T.; Hughes, M.; Nakagawa, T.; and Chelba, C. 2018. Denoising neural machine translation training with trusted data and online data selection. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, 133–143.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Wu, J.; Li, L.; and Wang, W. Y. 2018. Reinforced co-training. *arXiv preprint arXiv:1804.06035*.

Yang, Y.; Ma, Z.; Nie, F.; Chang, X.; and Hauptmann, A. G. 2015. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision* 113(2):113–127.

Zhang, X.; Kumar, G.; Khayrallah, H.; Murray, K.; Gwinnup, J.; Martindale, M. J.; McNamee, P.; Duh, K.; and Carpuat, M. 2018. An empirical exploration of curriculum learning for neural machine translation. *arXiv preprint arXiv:1811.00739*.