IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 August 2022, accepted 6 September 2022, date of publication 22 September 2022,
date of current version 30 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3208951

RESEARCH ARTICLE

Neural Machine Translation Transfer Model Based on Mutual Domain Guidance

YUPENG LIU, LEI ZHANG, AND YANAN ZHANG
Department of Software Engineering, Harbin University of Science and Technology, Harbin 150080, China

Corresponding author: Yupeng Liu (flyeagle99@126.com)

This work was supported in part by the National Natural Science Youth Foundation of China under Grant 61300115, in part by the China Postdoctoral Science Foundation of China under Grant 2014M561331, and in part by the Science and Technology Research Project of Heilongjiang Provincial Department of Education under Grant 12521073.

ABSTRACT The neural machine translation (NMT) model is a data hungry and domain-sensitive model but it is almost impossible to obtain a large number of labeled data for training it. This requires the use of domain transfer strategy. In order to solve the problem of domain data mismatch, this paper proposes a neural machine translation transfer model based on domain mutual guidance and establishes the continuous impact through the framework of mutual guidance. At the same time, self-ensemble and self-knowledge-distillation are used in these independent domains so that the model will not deviate from the domain too much. Furthermore, the model can better train the models from the batching way of domain data. It mainly uses the pretraining model out of domain, distillation of existing models in domain and data selection in the training process to guide the in-domain model. These are unified in the training framework, so that model training can be continuously and effectively guided in and out of domain. In this study, three typical experiment scenarios were comprehensive tested and our model was compared with many conventional classic methods. The experiment results showed that our proposed ''inter-domain transfer training'' and ''curriculum scheduling agent'' was effective and robust. The most important results and findings are that this comprehensive guided training framework (intra-domain and inter-domain) is suitable for the domain transfer in different scenarios, and this framework doesn't increase the decoding cost.

INDEX TERMS Neural machine translation, model transfer, self-ensemble, self-knowledge-distillation, curriculum learning.

I. INTRODUCTION

Transfer learning [1] is an important research topic in the machine learning field. Many machine learning methods can be considered to perform transfer learning, including multitask learning [2], domain adaptation [3], and semi-supervised learning [4]. Transfer learning mainly assumes two probability forms called domains ($\{P(x)|x \in X\}$, X for feature spaces) and tasks ($\{P(x|y)|x \in X, y \in Y\}$, Y for label spaces). The probability form of an entire model ($\{P(x, y)|x \in X, y \in Y\}$) can be obtained through the joint probability formula. Transfer learning is divided into source and target domains and tasks. $P_s(x)/P_s(y|x)$ can be used to represent the source domain/task, whereas $P_t(x)/P_t(y|x)$ can be used to represent the target domain/task. The transfer of learned information between domains is called domain adaptation. It can be applied to training and/or testing corpora from different domains. The transfer of learned information between different tasks is called multitask learning or system combination.

Neural machine translation (NMT) models have developed rapidly. Many translation frameworks have been reported in the relevant literature, including models using recurrent neural networks (RNN) [5], convolutional neural networks (CNN) [6], and transformer [7] models, as well as hybrid [8], [9] and simple frameworks for small devices (including neural architecture search [10] and knowledge distillation (KD) [11]). From the perspective of translation style, existing

The associate editor coordinating the review of this manuscript and approving it for publication was Tao Zhou.

methods can be divided into multilingual translation systems (including multiway translation such as one-to-many [12], many-to-one [13], and many-to-many [14], and multisource translation [15]), low-resource translation systems [16], and domain adaptation [17].

The proposed approach unifies in-domain and out-of-domain guidance to construct a training framework that continuously affects the training process. It combines the guidance within and out of the domain to construct a training framework. Unlike traditional methods, this framework can dynamically and continuously affect the training process so that the knowledge transfer relationship between domains can be considered more comprehensively. Training in the domain is not only related to in-domain and out-of-domain knowledge, but also to training data. To perform training and inference effectively, we consider various guidance signals from multiple perspectives, including in-domain and out-of-domain models and training data. The main contributions of this work are summarized here.

(1) A transfer training framework is proposed. It is designed to perform mutual guidance within and out of a given domain; both learning processes continuously guide each other to improve overall performance.
(2) A self-ensemble training objective based on self-knowledge-distillation and a dynamic pretraining are proposed. The self-ensemble training objective can better consider the in-domain information. The dynamic pretraining can better consider the out-of-domain information. During the training process, both in-domain and out-of-domain knowledge can be considered.
(3) An adaptive batching learning method based on reinforcement learning is proposed, which can better consider the sample feature. The difficulty of the training samples can be learned adaptively. Samples with different training difficulties are considered during the training process.

This paper discusses the application of deep networks in an NMT transfer task. In Section 2, related work is introduced. In Section 3, the neural translation model and objective function are introduced with a focus on the transformer structure and training objective of the transfer learning method. In Section 4, the training processes of model transfer and batching methods based on reinforcement learning are described. In Section 5, experimental results and their analysis are presented. Finally, in Section 6, our findings are summarized and some possible directions for further research are suggested.

II. RELATED WORK

A. DOMAIN TRANSFER

Domain transfer methods in machine translation are largely divided into data augmentation and model enhancement methods.

Data augmentation methods can be divided into monolingual corpus augmentation, synthetic bilingual corpus augmentation, and out-of-domain parallel corpora augmentation. Regarding monolingual corpus augmentation, Zhang & Zong, [17] used monolingual source data to enhance an NMT encoder through multitask learning to predict translated and reordered source monolingual data. Cheng *et al.*, [18] used NMT as an auto-encoder to reconstruct source and target monolingual data. Regarding synthetic bilingual corpus augmentation, a synthetic parallel corpus can be generated back translation of a target sentence to enhance the decoder [19]. Considering out-of-domain bilingual corpus augmentation, Chu *et al.*, [20] used tagged domain data to control their model. Chen & Huang [21] used a domain classifier to select sentences. Poncelas *et al.* [22] used sentence embedding to select sentences that were similar to the average in a given domain, but dissimilar to the average out of the domain. Poncelas *et al.* and Wang *et al.* [23], [24] used a feature attenuation algorithm to perform sentence selection. Cai *et al.* [70] and Vu *et al.* [71], both of whom used a cross-lingual neural network to retrieve target sentences with similar embeddings to a source sentence. Del *et al.* [72] clustered sentences embedding into domains that allowed for better adaptation than pre-defined corpora. Martins *et al.* [73] introduced a simple but effective caching strategy that avoided performing retrieval when similar contexts had been seen before.

Similarly, model enhancement methods may be largely divided into those performing training- (changing training objectives and processes), structure-, and decoding-based enhancement. Regarding training objectives, the basic concept is to weight training data. Shafiq *et al.* [25] used noise-contrastive estimation to perform domain adaptation of a translation system. Wang *et al.* [26] regularized translation tasks in the target domain using the marginal probability of the target side being monolingual. Chen *et al.* [27] trained a domain classifier to distinguish source sentences in the development set from those in the training set and the translation process was weighted. Wang *et al.* [28] used the loss difference between out-of-domain and in-domain information regarding bilingual sentence pairs to weight the translation process. For the training process, the basic concept is to fine-tune a developed model. Li *et al.* [29] only updated meta-parameters during the transfer process to achieve faster domain adaptation. Thompson *et al.* [30] analyzed the roles of an encoder, decoder, and probability calculation layer in feature transfer. Along these lines, Vilar [31] considered the contribution of a hidden layer in the transfer process. Regarding model-based enhancement, Gulcehre *et al.* [32] and Domhan & Hieber [33] integrated a language model (LM) and NMT into a decoder. Gulcehre *et al.* [32] trained models separately, whereas Domhan & Hieber [33] performed joint training. Britz *et al.* [34] used a discriminative network to distinguish domains [35]. Specifically, the contexts of domain-specific and domain-shared information were discriminated at the word level. Su *et al.* [36] used source context gates to accurately merge domain-independent and domain-shared contexts. Regarding decoding-based enhancement,

Dou *et al.* [37] used LMs within and out of a given domain to change the decoding target. Freitag *et al.* [38] used ensemble learning to integrate decoding objectives out of the target domain. Stickland *et al.* [74] decoupled these features between domains using adapter sub-network. Cao *et al.* [75] extended knowledge distillation for domain adaptation to a continual learning scenario. Pham *et al.* [76] proposed a method to better take advantage of these domain similarities, using a latent-variable model.

B. CURRICULUM LEARNING

In contrast to the traditional batch method [39], curriculum learning focuses on statistical efficiency rather than computational efficiency, even at the cost of performance. To the best of our knowledge, Tom & Ondrej [40] were the first to study curriculum learning in the context of NMT by using different features for curriculum scheduling on a Czech-English translation task. Curriculum learning is mainly designed to solve the problem of curriculum scheduling and can be divided into three research problems: sample difficulty, model capacity, and the relationship between them. Regarding sample difficulty, Xuan *et al.* [41] systematically modeled curriculum learning, classified sample difficulty, and conducted comprehensive experiments and analysis on the entire training process. Wang *et al.* [42] used sentence difficulty to weight sentence sampling. Wang *et al.* [43] weighted sentences in multiple domains to obtain the influence of sentences in a target domain. Regarding model capability, Platanios *et al.* [44] used a heuristic capacity function. Xu *et al.* [45] applied a BLEU-based process to a pretrained model as an adaptive capacity function. Liu *et al.* [46] used a norm to measure sample difficulty and model capacity. Zhou *et al.* [47] used data and model uncertainty to measure sample difficulty and model capacity. To accurately describe the relationship between sample difficulty and model capacity, Kumar *et al.* [48] and Zhao [49] expressed their nonlinear relationship and performed sequential decision modeling on different data selection methods. Wan *et al.* [50] used the concept of self-paced learning to weight an objective function to combine the training and sample selection processes. Wu *et al.* [77] used model uncertainty on a small, trusted multi-domain dataset to determine a curriculum across corpora. Hasler *et al.* [78] demonstrate that mixed fine-tuning without domain tags was complementary to directly regularizing parameters using Elastic Weight Consolidation (EWC).

III. TRANSLATION MODELING

A. MODEL

The proposed approach adopts a translation model based on a transformer [51] that relies on a self-attention network (a self-attention mechanism is an extension of traditional attention mechanisms [52]). The self-attention network is a type of neural network without recursion or convolutional operations that completely depends on a self-attention mechanism to implement an encoder (left side of Figure 1) and decoder (right side of Figure 1). The input sequence is token embedding and position embedding. The encoder is composed of N mixed add layers, multihead attention layers and feedforward layers, while the decoder is composed of N mixed add layers, multihead attention layers, feedforward layers and masked multihead attention layers. Unlike the multihead attention layer, the text sequence cannot be output at one time in masked multihead attention layers, and the ungenerated text is masked.

Specifically, a source input H^0 with position embedding is first transformed into a query matrix Q^0, key matrix K^0, and value matrix V^0. Multihead attention is then applied to Q^0, K^0, and V^0 as follows:

$$\text{MultiHead}(Q^0, K^0, V^0) = \text{Concat}(Q_1^1 : \cdots : Q_H^1)W^O, \quad (1)$$

$$Q_h^1 = \text{softmax}\left(\frac{Q_h^0 K_h^{0\text{T}}}{\sqrt{d}_{\text{mod } el}}\right)V_h^0, \quad (2)$$

$$Q_h^0, K_h^0, V_h^0 = Q^0 W_h^Q, K^0 W_h^K, V^0 W_h^V, \quad (3)$$

where Q_h^0, K_h^0, and V_h^0 are the h-th header query, key, and value matrices of layer zero, respectively. $\left\{W_h^Q, W_h^K, W_h^V\right\} \in \text{R}^{d_{model} \times d_k}$ represents the parameter matrix. d_{model} and d_k represent the dimensions of the model and header, respectively. A position-based feedforward neural network [a fully connected network with a rectified linear unit (ReLU) activation function applied to each position equally] is applied as follows:

$$Q^1 = \text{FFNN}(\text{MultiHead}(Q^0, K^0, V^0)) + Q^0, \quad (4)$$

where Q^1 is a source representation with global feature information and Q^0 is added to implement residual connections to overcome gradient vanishing.

Similarly, this processing sequence is formally expressed as a function f_{SAN}, which is used to learn the source representation Q^1.

$$Q^1 = f_{\text{SAN}}^1(Q^0, K^0, V^0). \quad (5)$$

The self-attention network uses a set of different layers to learn the source representation.

$$\left[Q^n = f_{\text{SAN}}^{\text{n}}(Q^{n-1}, K^{n-1}, V^{n-1})\right]_N. \quad (6)$$

Here, $[\cdots]_N$ ($n \in \{1, 2, \ldots, N\}$) indicates that N identical layers of the encoder are stacked together. The output Q_N of the N-th attention network layer is the final source representation that is sent to the decoder to learn a translation context vector used to predict the target word. The only difference between the decoder and encoder is the masked attention layer because the target output is generated dynamically.

The decoded output calculates the probability $p(y_j|y_{<j}, x; \Theta)$ of each word using the softmax function, where Θ denotes the parameter set of the encoder and decoder.

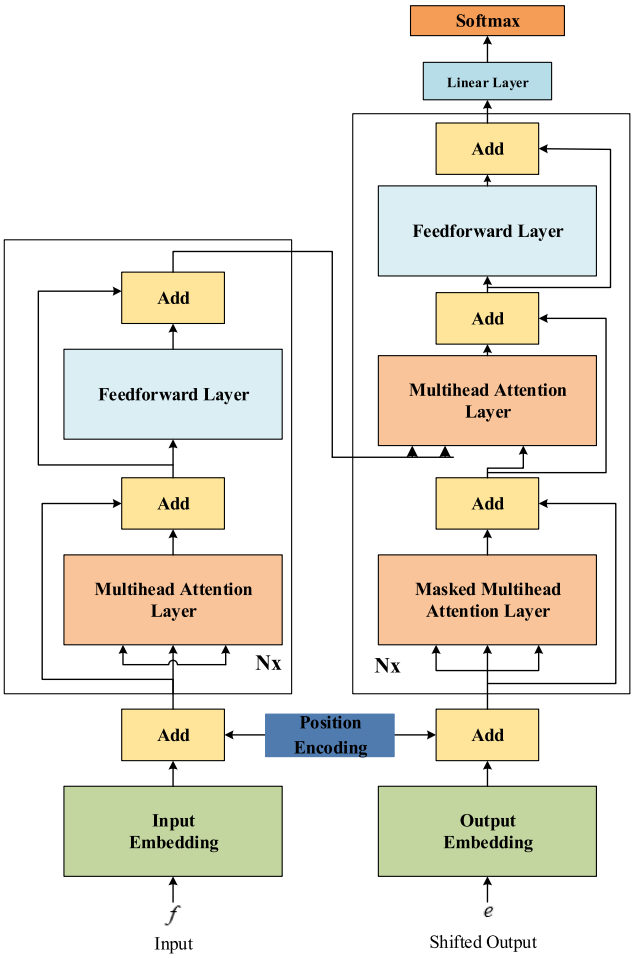

FIGURE 1. Neural translation model based on a transformer with encoder and decoder.

B. TRAINING OBJECTIVE

Because the proposed approach adopts the training mode of fine-tuning an out-of-domain model, in-domain knowledge is required to guide the training process and perform KD [53]. Distillation models typically include two submodels called the student and teacher models. The loss function of the student model is composed of the weighted sum of the following two components.

(1) The traditional loss function is a cross-entropy [54] one between prediction probability and true labels that is also known as a negative log-likelihood loss function.

$$L_{NNL}(\Theta_t; C) = \sum_{(e,f)\in C} \sum_{j=1}^{m} -I(y_j) \times \log p(y_j|y_{<j}, x; \Theta_t). \quad (7)$$

(2) The KD loss function is the cross-entropy or Kullback–Leibler distance [21] loss between the output probabilities of the student and teacher models.

$$L_{KD}(\Theta_t; C, \bar{\Theta}_t) = \sum_{(e,f)\in C} \sum_{j=1}^{m} -q(y_j|y_{<j}, \mathrm{x}; \bar{\Theta}_t) \times \log p(y_j|y_{<j}, x; \Theta_t). \quad (8)$$

Here, $q(y_j|y_{<j}, x; \bar{\Theta}_t)$ and $\bar{\Theta}_t$ are the distribution and parameter set of the teacher model, and $p(y_j|y_{<j}, x; \Theta_t)$ and Θ_t are the distribution and parameter set of the student model, respectively.

Inspired by Zeng *et al.* [53], a maximum strategy $\bar{\Theta}_t = \Theta_t^*$ (where Θ_t^* is the model parameter set yielding the best performance in previous rounds), average strategy $\bar{\Theta}_t = \frac{1}{N}\sum_n \Theta_t^{(n)}$ (average of the model parameter sets $\Theta_t^{(n)}$ in the previous n rounds), or weighted average strategy $\bar{\Theta}_t = \sum_n eval - norm(M(\Theta_t^{(n)}))\Theta_t^{(n)}$ [where $eval - norm(.)$ is the normalized function after the n-th evaluation $M(\Theta_t^{(n)})$ of the model parameter set $\Theta_t^{(n)}$] can be adopted for a self-ensemble model. In the average and weighted average strategies, the student model is typically more robust because it gathers information from the previous iterations of the teacher model.

IV. INTERDOMAIN TRANSFER TRAINING

Inspired by the concept of coteaching [55], the proposed approach models, i.e., the out-of-domain and in-domain models, perform transfer learning through pretraining.

Figure 2 illustrates the alternative training processes for in-domain and out-of-domain data. Each round of out-of-domain parameters is used to initialize the following round of in-domain parameters and vice versa. These processes are repeated to complete the mutual transmission of information. Through this alternative iterative training processes, each domain can absorb knowledge beneficial to this domain. Therefore, in-domain and out-of-domain features can be transferred to each other in a model-level transfer that can better retain shared knowledge between models and provides better transfer performance. Here, we need to evaluate the quality of the model. The source and target domain data C_s, C_t are divided into training sets C_s^{tr}, C_t^{tr} and development sets C_s^{val}, C_t^{val} that are used to train and evaluate the model, respectively.

Algorithm 1 is the domain transfer algorithm for the proposed NMT model, mainly divided into two stages.

(1) In the initialization phase, the main task is to complete initialization of the in-domain and out-of-domain model parameters.
- The $TrainModel(\cdot)$ function is used to train the model. The nondistillation objective function is used on the training set C_t^{tr} and the model parameter set $\Theta_t^{(0)}$ is initialized on $L_{NLL}(\Theta_t; C_t^{tr})$. The same process is performed for the source domain.

(2) In the iteration phase, the main task is to complete information transfer between the in-domain and out-of-domain models.
- The $TransModel(\cdot)$ function is used for model transfer. The objective functions with self-knowledge-distillation functions $L_{NLL}(\Theta_s^{(k-1)}; C_t^{tr})$ and $L_{KD}(\Theta_s^{(k-1)}; C_t^{tr}, \bar{\Theta}_t)$ are used in the training set C_t^{tr} (the weighted average method is used to balance the likelihood and distillation functions). To perform model transfer, the in-domain

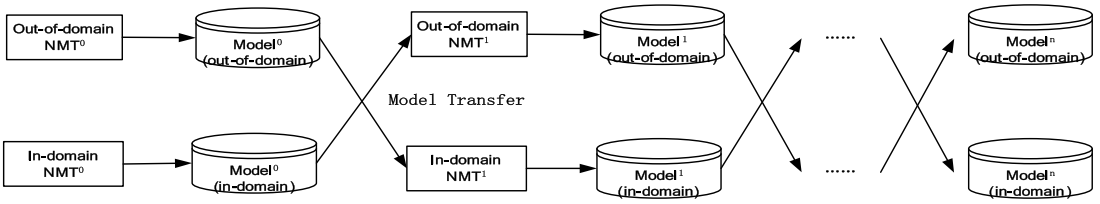

FIGURE 2. Alternative iterative training processes for in-domain and out-of-domain learning.

Algorithm 1 Model Transfer for NMT

Input: Training sets $\{C_t^{tr}, C_s^{tr}\}$, development sets $\{C_t^{val}, C_s^{val}\}$, and iteration number K
Output: In-domain NMT model$\hat{\Theta}_t$ and out-of-domain NMT model$\hat{\Theta}_s$
//In-domain model training
$\Theta_t^{(0)} \leftarrow \text{TrainModel}\left(L_{NLL}(\Theta_t; C_t^{tr})\right)$
//Out-of-domain model training
$\Theta_s^{(0)} \leftarrow \text{TrainModel}\left(L_{NLL}(\Theta_s; C_s^{tr})\right)$
//Initializing in-domain and out-of-domain self-ensemble model parameters
$\bar{\Theta}_t \leftarrow \Theta_t^{(0)}, \bar{\Theta}_s \leftarrow \Theta_s^{(0)}$
for $k = 1, 2, \ldots, K$ **do**
  //In-domain model transfer training and evaluation
$\Theta_t^{(k)} \leftarrow \text{TransModel}\left(L_{NLL}(\Theta_s^{(k-1)}; C_t^{tr}), L_{KD}(\Theta_s^{(k-1)}; C_t^{tr}, \bar{\Theta}_t)\right)$
$\bar{\Theta}_t \leftarrow \text{EvalModel}\left(C_t^{val}, \Theta_t^{(k)}\right)$
  //Out-of-domain model transfer training and evaluation
$\Theta_s^{(k)} \leftarrow \text{TransModel}\left(L_{NLL}(\Theta_t^{(k-1)}; C_s^{tr}), L_{KD}(\Theta_t^{(k-1)}; C_s^{tr}, \bar{\Theta}_s)\right)$
$\bar{\Theta}_s \leftarrow \text{EvalModel}\left(C_s^{val}; \Theta_s^{(k)}\right)$
end for

model parameter set Θ_t in a given round is initialized using the previous round of the out-of-domain model parameter set $\Theta_s^{(k-1)}$. After initialization, fine-tuning based on pretraining is performed on the in-domain model. The same processes are repeated for the source domain.

- The *EvalModel*$(\cdot)$ and self-ensemble functions are used to evaluate the performance of $\Theta_t^{(k)}$ on the development set C_t^{val} and form the self-ensemble parameter set $\bar{\Theta}_t$ (see Section III.B).

V. TRAINING PROCESS IN DOMAIN

For the in-domain data, we aim to select each batch of training data to enable the model to learn effectively. The basic concept is to adopt the method of curriculum learning [42], [44], [45], [47]. The main advantage of this method is that it can be planned in batches (learning the simple samples before the difficult ones). However, such methods use heuristics to define the relationship between the accuracy and uncertainty of the model, and the difficulty and uncertainty of the original sample. Because the batch process is a sequential decision-making process (the data for training are selected according to the capacity of the model), we adopt the reinforcement learning framework to adaptively learn the relationships between these characteristics.

A. REINFORCEMENT MODEL REPRESENTATION

Curriculum scheduling can be considered as a Markov decision process (MDP), where an agent interacts with its environment (data and NMT model) to perform sample selection. An MDP can be defined by a five tuple (S, A, P, R, γ).

- **State** S. A neural translation model and data are used to represent the current state s. The difficulty of the present round of training is described by a feature. Generally, there are two types of difficulty [41]: model- and linguistics-based difficulty.
- **Action** A. Action $a \in R^k$ is a continuous parameter vector, meaning that it selects samples according to the current training scenario to perform curriculum scheduling.
- **Transition** P. Once it is determined whether the current sample is in the current curriculum or not, the state transition probability $p(s'|s, a)$ is confirmed.
- **Reward** R. Given a state and action, the scheduling agent will provide an immediate reward $(r(s, a))$ according to the current training scenario.
- **Discount Rate** γ. $\gamma \in [0, 1]$ is a discount factor that measures the current value of long-term rewards.

The entire reinforcement process can be described as follows. In each time step, the scheduling agent presents $a \in A$ according to the current data and model state $s \in S$, and obtains the corresponding reward $r(s, a)$. The original state updates to the new state s' according to the state transition probability $p(s'|s, a)$. The goal of the scheduling agent is to identify the optimal strategy $(\mu_\phi : S \times A \mapsto [0, 1])$ to maximize the expected cumulative reward. In the proposed approach, we adopt the classic deep deterministic policy gradient (DDPG) algorithm. The DDPG algorithm utilizes an actor-critic framework that can model continuous behavior. Compared with a model based solely on actors such as the REINFORCE algorithm [56], the existence of the critic reduces the update variance and accelerates the convergence of the model.

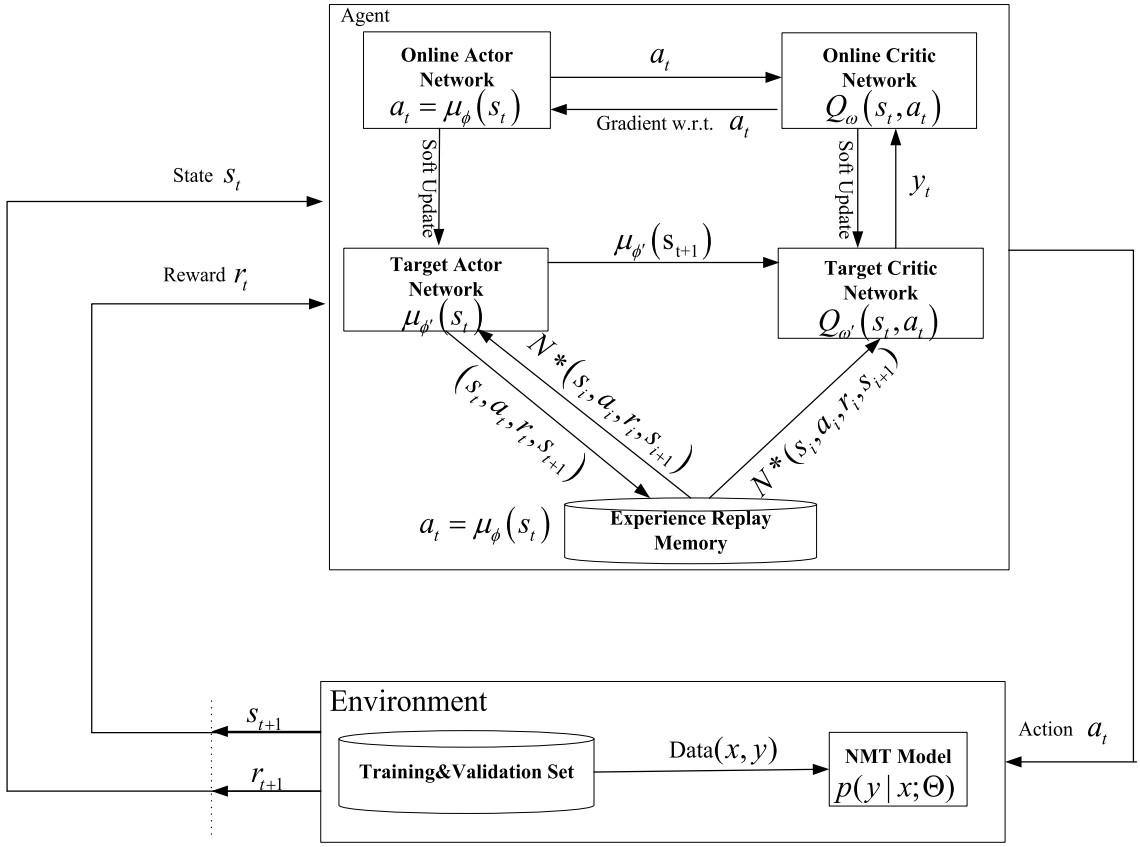

FIGURE 3. Schematic of curriculum scheduling based on deep deterministic policy gradient.

Figure 3 presents a schematic of the DDPG algorithm for curriculum scheduling. Take the interaction between the agent and the environment at time t as an example (the action, the state and the reward at time t are denoted as a_t, s_t and r_t). The environment contains machine translation model $p(y|x; \Theta)$, training and validation data $data(x, y)$. The agent is divided into five components, including online actor-critic ($a_t = \mu_\phi(s_t)$ and $Q_\omega(s_t, a_t)$) and target actor-critic networks ($\mu_{\phi'}(s_t)$ and $Q_{\omega'}(s_t, a_t)$), and experience replay memory. The critic networks are divided into online and target networks because if the target network is updated frequently, the learning process becomes highly unstable. The main purpose of experience replay memory is to overcome the correlation data and nonstationary distribution of experience data by storing the training sample (s_t, a_t, r_t, s_{t+1}) at the current time t and taking out N training samples $N * (s_i, a_i, r_i, s_{i+1})$. The curriculum scheduling agent needs to exploit and explore the environment to identify better strategies. To induce the model to perform exploration more effectively, we introduce random noise on the action strategy.

B. AGENT STRUCTURE

The basic structure of the actor–critic network, shown in Figure 4, comprises several parts. Figure 5 is a detailed diagram of FM and SRM in Figure 4.

- Feature model (FM in figure 4): The data and model cannot be described comprehensively. Here, we take the representative data and model description as the features, of which there are two types. Features in FM include:

➢ Data features (or linguistic features [41]): (1) source/target sentence length L_x and L_y [40]; (2) source/target n-gram sparsity feature [41], [57] that calculates the frequency of the word n-gram in a sentence($n = \{1, 2, 3, 4\}$); (3) source/target-side uncertainty $u(x)$ and $u(y)$ [47] that uses a pretrained LM to compute sentence complexity. Additionally, sentence embedding features are used to obtain a more refined data representation.

➢ Model features include a log-likelihood score. Additionally, to represent the model more finely and reduce its complexity, the output vector of the last softmax layer in the transformer is used to represent the model embedding features.

From different perspectives, these features can be divided into different types: deep features (such as sentence and model embedding features) and traditional (other) features, as well as model and data features. To process these low-order data and model features effectively to obtain high-order features, the features are further divided into four types according to the aforementioned feature types: a traditional model feature

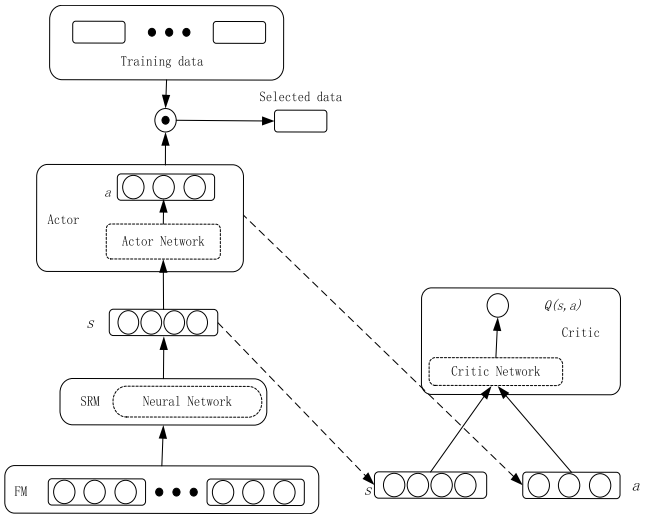

FIGURE 4. Schematic of curriculum scheduling agent (actor-critic) network.

f_{tm} (log-likelihood score), traditional data feature f_{td} (representing sentence length, sparsity, and uncertainty), deep model feature f_{dm} (model embedding feature), and deep data feature f_{dd} (sentence embedding feature). Because the dimensions of these features differ, a simple linear transformation is used to map them to the same dimension $e_i \in \mathrm{R}^{1\times n}$, $i \in \{tm, td, dm, dd\}$, as indicated by the blue lines in Figure 5.

- State representation model (SRM in figure 4): The state representation can be obtained through interactions between features. Here, the concept of simple pairwise interaction is adopted. As indicated by the red lines in Figure 5 (element-wise Hadamard product $\otimes$ between vectors), the state representation $s \in \mathrm{R}^{1\times 12n}$ is obtained.
- Actor network: By using two ReLU layers, i.e., a tanh layer and softmax layer, the state representation s is transformed into an action $a = \mu_\phi(s)$ as the output of the critic network. Specifically, the action a is defined as a continuous probability parameter vector to select samples. Because the output action is the input of the critic network, the actor network will update this parameter according to the direction of accelerating Q-values.
- Critic network: This model is a deep Q-network that brings the parameterized deep neural network $Q_w(s, a)$ closer to the real state action function $Q^\pi(s, a)$ (i.e., a Q-value function)

C. AGENT TRAINING ALGORITHM

Algorithm 2 summarizes the training process of the curriculum scheduling agent. Specifically, at time t, the training process includes two stages: transition generation (lines 5 to 9) and updating the model (lines 10 to 16).

- In the first stage, the scheduling agent generates a state through the SRM (a state phasor model), then uses an ε-greedy function to explore the generated action and calculate the reward function, and finally stores the result in the experience replay memory.

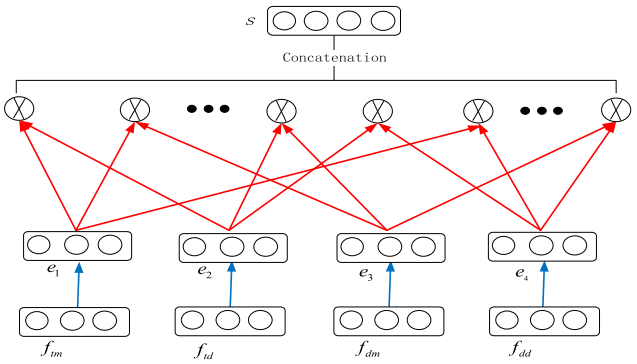

FIGURE 5. Schematic of Feature Model (FM) and State Representation Model (SRM).

- In the second stage, some samples are removed from the experience replay memory to update the actor and critic networks for the two objective functions. According to the time difference learning method [58], the objective function and update formula for the critic network are expressed as follows:

$$J(Q_\omega) \approx \frac{1}{N}\sum_i \left(y_i - Q_\omega\left(s_i, a_i\right)\right)^2$$
$$\text{with } y_i = r_i + \gamma Q_{\omega'}\left(s_{i+1}, \mu_{\phi'}\left(s_{i+1}\right)\right), \quad (9)$$
$$\omega' = \omega + \alpha_\omega \delta \nabla_a Q_\omega\left(s, a\right), \quad (10)$$

where $\delta = y_i - Q_\omega\left(s_i, a_i\right)$ is the temporal difference. According to the deterministic policy gradient theorem [59], the objective function and update formula for the actor network are expressed as follows:

$$J(\mu_\phi) \approx \frac{1}{N}\sum_t Q_\omega\left(s, a\right)\big|_{s=s_t, a=\mu_\phi(s_t)}, \quad (11)$$
$$\phi' = \phi + \alpha_\phi \nabla_a Q_\omega\left(s, a\right)\Big|^{s=s_t, a=\mu_\phi(s_t)}$$
$$\times \nabla_\phi \mu_\phi\left(s\right)\big|_{s=s_t}, \quad (12)$$

where α_ω and α_ϕ are the learning rates of the two networks.

VI. EXPERIMENTS AND ANALYSIS

A. EXPERIMENTAL SETUP

1) DATASET

We performed an experiment on a Chinese-English translation task. We filtered sentences that were longer than 50 characters. For the English data, we used the Moses script[1] to perform word segmentation and identify lowercase letters. For the Chinese data, we used the Stanford word segmentation tool[2] to perform word segmentation. Byte-pair encoding [60] was used to segment sentences in parallel corpora (the number of merging operations was 16k). To verify the effectiveness of the proposed approach, we conducted experiments under three data scenarios.

(1) In-domain and out-of-domain (general domain) setting: OpenSubtitles 2018 [61] and WMT 2017 [62]were used as out-of-domain data and randomly divided into a training dataset (1.8m in total), development set (2k in total),

Algorithm 2 Training Algorithm for the Curriculum Scheduling Agent

Initialize actor μ_ϕ and critic Q_ω with parameters ϕ and ω
Initialize target network $\mu_{\phi'}$ and $Q_{\omega'}$ with weights $\phi' \leftarrow \phi$ and $\omega' \leftarrow \omega$
Initialize experience replay memory M and soft update parameter τ
for $k = 1 \ldots K$ **do**
  for *number of RL training iterations* **do**
    Observe state s_t
    Obtain action a_t according to policy μ_ϕ with ε-greedy exploration
    Update $p(y|x; \Theta^{(k)})$ with selected sample to obtain $p(y|x; \Theta^{(k')})$
    Calculate the perplexity difference on the validation set C^{val} between $p(y|x; \Theta^{(k')})$ and $p(y|x; \Theta^{(k)})$ as r_{t_z}
    Observe new state s_{t+1} and store transition (s_t, a_t, r_t, s_{t+1}) in M
    Sample mini-batch transitions $N * (s_i, a_i, r_i, s_{i+1})$ in M using the prioritized experience replay sampling technique
    Update the critic network through Equation (11)
    Update the actor network through Equation (13)
    Update the parameters in the SRM with backpropagation from the update signal of the actor
    Update the target networks:

$$\mu' = \tau\mu + (1-\tau)\,\mu'$$
$$\omega' = \tau\omega + (1-\tau)\,\omega'$$

    Select data $C^{tr'}$ from C^{tr} using μ_ϕ
    Update $p(y|x; \Theta^{(k)})$ with $C^{tr'}$ to obtain $p(y|x; \Theta^{(k+1)})$

and testing set (2k in total), denoted by the tag "GEN." The in-domain data were the same as those in the specific domain setting.

(2) Specific domain setting: We used the UM corpus [63] divided into specific domains. Three common domains were selected: news (denoted as NEWS, 1.5m in total), education (denoted as EDU, 1.5m in total), and law (denoted as LAW, 1.5m in total). We randomly selected 2000 sentences from each domain as the development set and 2000 sentences as the test set.

(3) Low-resource domain setting: This setting was the same as the in-domain and out-of-domain scenarios, but 50k,

(4) 100k, and 300k data samples were randomly selected from the training set in the domain.

2) NEURAL NETWORK

We used PyTorch[3] to implement the baseline model and our transformer model. The same hyperparameter settings were used for the self-ensemble and proposed models. L2-norm regularization was used to mitigate overfitting. Both the word vector and hidden variable length were set to 512. All parameters were initialized using a uniform distribution in $[-0.1, 0.1]$. Our model used the Adam algorithm as an optimizer and the initial learning rate was set to 0.0005. When the performance of the development set did not exceed that of the previous eight rounds of checkpoints, the learning rate was set to 0.8 times the original value. When the performance of the development set did not improve in 20 rounds of checkpoints, the training process was terminated (one checkpoint was equivalent to 1000 updates).

3) REINFORCEMENT LEARNING

For the curriculum scheduling agent, the actor-critic framework was adopted and the system was constructed based on reference [64]. The experience replay memory size was set to 2500 and a warm-up phase of 500 steps was performed. The mixing factor of the target and online networks was $\tau = 0.1$. The target network was updated at 100 steps. The discount factor was $\gamma = 0.99$.

4) MACHINE TRANSLATION

The source and target uncertainties were calculated by training the four-gram LM using KenLM [65] with modified Kneser–Ney smoothing. The evaluation criteria were based on four-gram BLEU scores [66]. The proposed model is referred to as the domain mutual guidance model (DMGM). A classic baseline system was selected for comparison.

- Simple model: **IN**: Only data in the domain were used to perform training; **IN** + **OUT**: Data in both domains were mixed together.
- Fine-tuning model: Oversampling (**OS**) [67] was first used to train the NMT model on the out-of-domain training corpus and then fine-tune the NMT model. Iterative dual domain adaptation (**IDDA**) [53] was alternately used to fine-tune the out-of-domain and in-domain data.
- Discriminative model: Discriminative mixing (**DM**) [34] included a discriminator in the encoder to predict the domain of translation. For the discriminator, the encoder and decoder were optimized jointly.
- Domain label model: Domain control (**DC**) [68] added an additional domain label to each source statement to merge the domain information with the source sentences. Target token mixing (**TTM**) [34] was used to mark the domain of the target-side sequence.
- KD model: **KD** [69] was used to train a model in its own domain and fine-tune the out-of-domain model using in-domain data. The trained model was then used in domain for supervision.

B. OVERALL PERFORMANCE

In our reported results, boldface indicates the best performance on the testing set. A significance test was conducted (where ** indicates significance level $\alpha = 0.01$, and * indicates significance level $\alpha = 0.05$). GEN-NEWS indicates that the out-of-domain and in-domain data came from the general

TABLE 1. Performance of in-domain and out-of -domain scenario and specific domain experiment scenar.

Model	In-domain and out-of-domain scenario			Specific domain scenario		
	GEN-NEWS	GEN-EDU	GEN-LAW	NEWS-EDU	NEWS-LAW	EDU-LAW
Simple model						
IN	31.42 (23.77)	30.62 (22.78)	30.25 (22.61)	32.25 (30.63)	32.62 (30.14)	30.25 (30.21)
IN+OUT	29.82 (21.46)	28.77 (20.85)	28.46 (20.17)	30.84 (29.53)	30.41 (29.02)	29.41 (29.14)
Fine-tuning model						
OS	33.65 (25.75)	32.78 (24.65)	32.63 (24.98)	34.45 (32.85)	34.54 (32.65)	32.56 (32.47)
IDDA	34.10 (26.42)	33.28 (25.47)	33.14 (25.20)	35.15 (33.56)	35.28 (33.71)	33.61 (33.87)
Discriminative model						
DM	32.31 (24.74)	31.35 (23.46)	31.28 (23.84)	33.41 (31.88)	33.54 (31.66)	31.77 (31.05)
Domain token model						
DC	32.26 (24.33)	31.75 (23.36)	31.14 (23.21)	33.35 (31.25)	33.14 (31.14)	31.35 (31.21)
TTM	32.54 (24.71)	31.57 (23.45)	31.48 (23.62)	33.74 (31.56)	33.05 (31.47)	31.58 (31.51)
KD model						
KD	32.61 (24.42)	31.54 (23.28)	31.45 (23.54)	33.68 (31.93)	33.09 (31.66)	31.58 (31.44)
DMGM						
DMGM	**35.52**** (**27.42**)	**34.91**** (**26.65**)	**34.88*** (**26.88***)	**36.12**** (**34.59**)	**36.41**** (**34.25**)	**34.18*** (**34.15***)

and news domains, respectively. The other symbols are used similarly.

Table 1 lists the experimental performance measures under in-domain, out-of-domain, and specific-domain scenarios. An analysis of Table 1 reveals that the cross-domain transfer method is able to improve performance to a certain extent. For the overall baseline model, the performance of the simple model is the worst (one possible reason is that it does not make good use of domain knowledge). Additionally, the discriminative, domain label, and KD models exhibit the second-best performance, which can be attributed to their separate use of out-of-domain and in-domain knowledge. The fine-tuning model exhibits the best performance, which may be because the out-of domain and in-domain corpora are used synthetically. However, this method has no effective guidance compared to the proposed approach. From the perspective of the baseline classification model, the method using mixed corpora (IN+OUT) in the simple model does not perform as well as the method using only in-domain corpora (IN). IDDA performs better than the OS model (one possible reason is that the iterative method is able to integrate out-of-domain and in-domain knowledge more effectively). When comparing the in-domain and out-of-domain scenarios to the specific-domain scenario, one can see that the performance of the specific domain is better than that of the general domain, which can be attributed to the fact that the general domain contains information from multiple domains, causing the data to be relatively noisy. From the perspective of the experimental scenarios in and out of the domain, the transfer effect of the news corpus and general domain is better. One possible reason for this is that the news corpus has stronger generalization ability in the domain. From the perspective of the specific-domain experimental scenarios, we can make the same observation.

Table 2 presents the performance of the low-resource experimental scenario. NEWS-50 refers to the use of 50k in-domain samples and the other results are similarly named. As a result of the mismatching between the amount of out-of-domain and low-resource-domain data, the fine-tuning model, KD method, and our method only adopt a single approach, namely, using data from out of the domain for initialization, and they do not iteratively improve each other. Additionally, the in-domain data are oversampled in the discriminant and domain label models. As a result of this in-domain-oriented training method, this section only discusses the in-domain performance.

For the overall and classification baseline models (various systems are compared in Table 2), the conclusions are similar

TABLE 2. Performance in the low-resource experimental scenario.

Model	NEWS-50	NEWS-100	NEWS-300	EDU-50	EDU-100	EDU-300	LAW-50	LAW-100	LAW-300
Simple model									
IN	9.21	17.56	23.88	9.28	17.47	23.45	8.45	16.25	22.98
IN+OUT	6.01	13.22	18.18	6.12	13.22	18.18	5.89	12.27	17.76
Fine-tuning model									
OS	10.33	18.64	24.92	9.88	17.98	24.12	8.98	17.21	23.12
Discriminative model									
DM	10.51	18.27	24.62	9.24	17.58	24.21	8.45	17.34	23.24
Domain token model									
DC	10.52	18.86	24.57	9.75	17.67	24.31	8.52	17.71	23.51
TTM	10.64	18.58	24.64	9.64	17.25	24.45	8.87	17.42	23.12
KD model									
KD	10.28	18.47	24.27	9.74	17.68	24.17	8.69	17.14	23.34
Domain mutual guidance model									
DMGM	**12.17****	**19.21****	**25.34****	**11.87***	**19.01***	**25.12****	**10.95***	**18.62***	**24.78***

TABLE 3. Influence of integration modes on performance.

Scenario	Dataset	Transformer+1	Transformer+2	Transformer+3
In-domain and out-of-domain scenario	GEN-NEWS	35.24±0.29 (27.14±0.42)	35.21±0.22 (27.13±0.21)	**35.52±0.18** (**27.42**±0.21)
	GEN-EDU	34.41±0.32 (25.21±0.34)	34.68±0.23 (26.58±0.22)	**34.91±0.17** (**26.65±0.24**)
	GEN-LAW	34.54±0.34 (26.88±0.40)	**34.93±0.17** (26.62±**0.16**)	34.88±0.24 (**26.88**±0.19)
Specific-domain scenario	NEWS-EDU	36.12±0.29 (33.89±0.32)	**36.24±0.24** (34.42±**0.17**)	36.12±0.26 (**34.59**±0.24)
	NEWS-LAW	36.09±0.38 (33.82±0.29)	36.38±**0.27** (**34.64±0.24**)	**36.41**±0.28 (34.25±0.27)
	EDU-LAW	33.28±0.40 (33.17±0.31)	**34.24**±0.21 (**34.25±0.18**)	34.18±**0.18** (34.15±0.24)
Low-resource scenario	NEWS-50	11.62±0.33	**12.18±0.21**	12.17±0.24
	NEWS-100	18.98±0.32	**19.24±0.28**	19.21±0.32
	NEWS-300	25.17±0.34	25.21±0.26	**25.34±0.17**
	EDU-50	11.42±0.41	11.67±0.19	**11.87±0.16**
	EDU-100	18.83±0.32	**19.09±0.18**	19.01±0.21
	EDU-300	24.78±0.35	**25.25±0.13**	25.12±0.22
	LAW-50	10.62±0.34	10.65±0.32	**10.95±0.21**
	LAW-100	17.82±0.40	**18.74±0.12**	18.62±0.24
	LAW-300	24.08±0.29	24.64±0.28	**24.78±0.19**

to those in the resource-rich scenarios. The best performance can be observed for the news data and when comparing various domains, the performance of the corpus increases significantly for the news data (which demonstrates the particularity of low-resource scenarios). When comparing low-resource (Table 2) and resource-rich methods (Table 1), the performance of low-resource methods is much worse. It should be highlighted that the performance of the hybrid corpus (IN+OUT) is significantly worse than that of the in-domain corpus (IN). One possible reason is that the noise in the hybrid corpus has a greater influence in the low-resource scenario.

C. INFLUENCE OF THE SELF-ENSEMBLE MODE

Table 3 presents the effects of different self-ensemble methods on performance. The maximum, average, and weighted strategies are represented by symbols +1, +2, and +3,

TABLE 4. Ablation experiments with different characteristics.

Model	In-domain and out-of-domain scenario GEN-NEWS	Specific-domain scenario NEWS-EDU	Low-resource scenario NEWS-100
Base	32.84（24.17）	33.21（31.28）	14.81
Traditional data features			
SenLen	33.02（24.82）	33.62（31.81）	15.21
n-gram	33.24（25.01）	33.74（32.07）	15.42
Uncertainty	33.62（25.41）	33.99（32.43）	15.82
Traditional model features			
Logp	33.81（25.62）	34.24（33.12）	16.71
Deep data features			
SenEmb	34.62（26.58）	35.15（33.99）	18.03
Deep model features			
ModelEmb	**35.52（27.42）**	**36.12（34.59）**	**19.21**

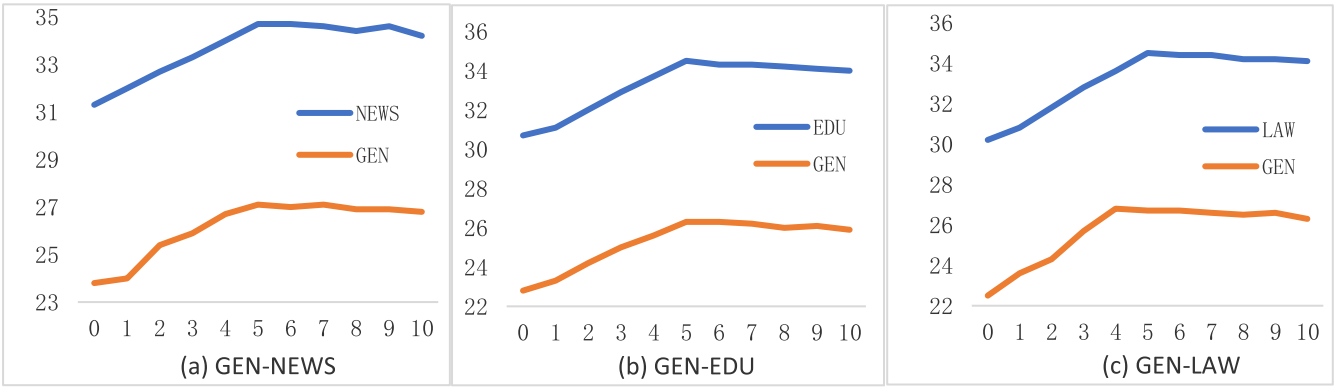

FIGURE 6. Effect of iterations in the in-domain and out-of-domain scenario.

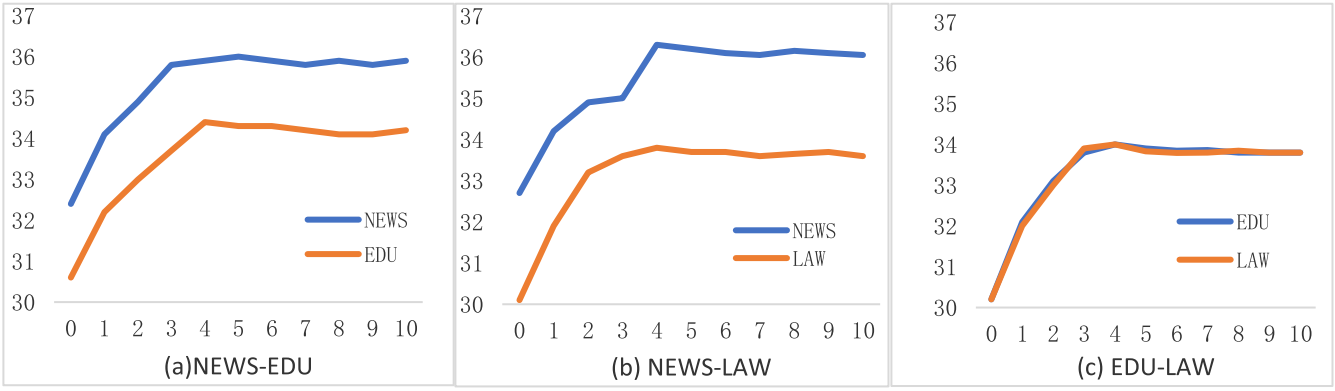

FIGURE 7. Effect of iterations in the specific-domain scenario.

respectively. The performances of different self-ensemble methods are approximately the same. One possible reason is that as model transfer proceeds, the knowledge obtained by distillation tends to be maximized. The boldface font indicates the best average performance and smallest variation range. Compared to the maximum strategy, the average and weighted strategies exhibit better robustness.

D. INFLUENCE OF STATE FEATURES

In Table 4, representative datasets for three scenarios are highlighted to present the results of our experiments. The other datasets yielded similar conclusions. **Base** is the result of learning without using curriculum scheduling. **SenLen, *n*-gram, Uncertainty, Logp, SenEmb,** and **ModelEmb** are the results of using the source and target sentence length,

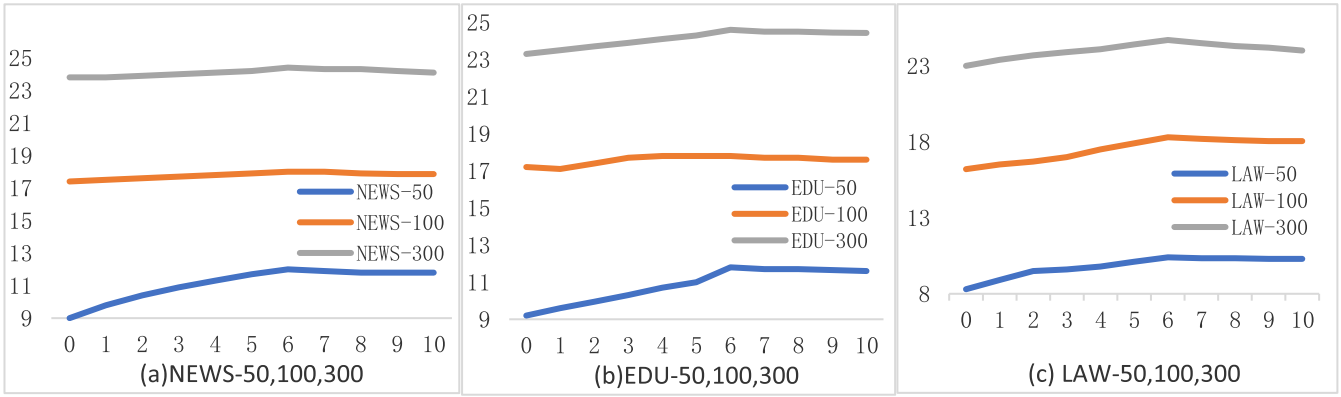

FIGURE 8. Effect of iterations in the low-resource scenario.

source/target *n*-gram sparsity, uncertainty, log-likelihood, sentence embedding and model embedding features, respectively. Compared with the baseline model, the performance improved significantly for the proposed model, particularly in the low-resource scenarios (one possible reason is that the data distribution is sparse and sensitive to the data). The deep features exhibit better performance than the traditional features and reflect the overall context of the target sentences.

E. ITERATION NUMBER SENSITIVITY

The number of alternative iterative training is a key factor affecting the effectiveness of information transmission between domains. In the framework of inter-domain transfer, the iteration number k is a key hyperparameter that directly determines the number of translated knowledge transfers. We varied k from 0 to 10 in steps of 1. When $k = 0$, the framework degenerated to a single-domain model.

In the in-domain and out-of-domain (Figures 6(a) to 6(c)), specific-domain (Figures 7(a) to 7(c)), and low-resource (Figures 8(a) to 8(c)) scenarios, DMGM exhibited performance near the top in the sixth, fourth, and seventh iterations, respectively, which could be attributed to the special data distributions in the general and small-data domains. When comparing Figures 6 and 7, one can observe that the two domains converged together in the in-domain and out-of-domain scenario as well as in the specific-domain scenario, indicating that the two domains were able to learn from each other until they were trained well. In each graph (e.g., when comparing Figures 6(a) to 6(c)), the convergence times of different domains were largely similar, indicating that the framework was not particularly sensitive to the domains. Naturally, this framework can also be applied to other domain transfer tasks.

VII. CONCLUSION

This paper explores the application of transfer learning in classical neural translation model. It proposes an effective translation transfer method that mainly includes: (1) the framework of alternative training out of and in domain, so that the in-domain and out-of-domain knowledge can be transferred to each other; (2) in the in-domain, the previous knowledge can be used to guide the training of the current model through self-ensemble and distillation; (3) the difficulty of data will also be considered in the current model training. The classical transformer model was used to analyze the experiment results in three typical experimental scenarios, thereby, showing the effectiveness and robustness of the proposed model.

From the commercial point of view, this framework is only on the training stage and has no impact on the decoding stage. It is suitable for offline training and deployment to online system. In the future, we hope to develop a transfer training framework suitable for more domains and further reduce the training cost. In addition to cross-domain, we hope that this method can also be applied to other similar tasks, such as cross-lingually etc.

REFERENCES

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2009.

[2] R. A. Caruana, "Multitask learning: A knowledge-based source of inductive bias," in *Proc. 10th Int. Conf. Mach. Learn.*, 1993, vol. 10, no. 1, pp. 41–48.

[3] H. Daumé, "Frustratingly easy domain adaptation," 2007, *arXiv:0907.1815*.

[4] Z. Xu and I. King, *Introduction to Semi-Supervised Learning*. San Rafael, CA, USA: Morgan & Claypool, 2009.

[5] Y. Wu, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016, *arXiv:1609.08144*.

[6] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1243–1252. [Online]. Available: http://dl.acm.org/citation.cfm?id=3305381.3305510

[7] A. Vaswani *et al.*, "Attention is all you need," 2017, *arXiv:1706.03762*.

[8] L. Miculicich, N. Pappas, D. Ram, and A. Popescu-Belis, "Self-attentive residual decoder for neural machine translation," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, vol. 1. New Orleans, LA, USA: Association for Computational Linguistics, 2018, pp. 1366–1379. [Online]. Available: https://www.aclweb.org/anthology/N18-1124, doi: 10.18653/v1/N18-1124.

[9] C. Wang, M. Li, and A. J. Smola, "Language models with transformers," 2019, *arXiv:1904.09408*.

[10] D. R. So, C. Liang, and Q. V. Le, "The evolved transformer," 2019, *arXiv:1901.11117*.

[11] D. Zhang, J. Crego, and J. Senellart, "Analyzing knowledge distillation in neural machine translation," in *Proc. Int. Workshop Spoken Lang. Transl. (IWSLT)*, 2018, pp. 23–30.

[12] D. Dong, H. Wu, W. He, D. Yu, and H. Wang, "Multi-task learning for multiple language translation," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, vol. 1, 2015, pp. 1723–1732, doi: 10.3115/v1/P15-1166.

[13] J. Lee, K. Cho, and T. Hofmann, "Fully character-level neural machine translation without explicit segmentation," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 365–378, Dec. 2017. [Online]. Available: http://aclweb.org/anthology/.Q17-1026

[14] O. Firat, K. Cho, and Y. Bengio, "Multi-way, multilingual neural machine translation with a shared attention mechanism," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2016, pp. 866–875, doi: 10.18653/v1/N16-1101.

[15] B. Zoph and K. Knight, "Multi-source neural translation," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2016, pp. 30–34, doi: 10.18653/v1/N16-1004.

[16] R. Sennrich and B. Zhang, "Revisiting low-resource neural machine translation: A case study," 2019, *arXiv:1905.11901*.

[17] J. Zhang and C. Zong, "Exploiting source-side monolingual data in neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 1535–1545.

[18] Y. Cheng, W. Xu, Z. He, W. He, H. Wu, M. Sun, and Y. Liu, "Semi-supervised learning for neural machine translation," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2016, pp. 1965–1974.

[19] R. Sennrich, B. Haddow, and A. Birch, "Improving neural machine translation models with monolingual data," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, vol. 1, 2016, pp. 86–96.

[20] C. Chu, R. Dabre, and S. Kurohashi, "An empirical comparison of domain adaptation methods for neural machine translation," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, Vancouver, BC, Canada, 2017, pp. 1–7.

[21] B. Chen and F. Huang, "Semi-supervised convolutional networks for translation adaptation with tiny amount of in-domain data," in *Proc. 20th SIGNLL Conf. Comput. Natural Lang. Learn.*, 2016, pp. 314–323.

[22] A. Poncelas, G. M. de Buy Wenniger, and A. Way, "Feature decay algorithms for neural machine translation," in *Proc. 21st Annu. Conf. Eur. Assoc. Mach. Transl.*, 2018, pp. 239–248.

[23] A. Poncelas, A. Way, and K. Sarasola, "The ADAPT system description for the IWSLT 2018 Basque to English translation task," in *Proc. 15th Int. Workshop Spoken Lang. Transl. (IWSLT)*, 2018, pp. 76–82.

[24] R. Wang, A. Finch, M. Utiyama, and E. Sumita, "Sentence embedding for neural machine translation domain adaptation," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, 2017, pp. 1–7.

[25] S. Joty, H. Sajjad, N. Durrani, K. Al-Mannai, A. Abdelali, and S. Vogel, "How to avoid unwanted pregnancies: Domain adaptation using neural network models," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Lisbon, Portugal, 2015, pp. 1–12.

[26] Y. Wang, Y. Xia, Z. Li, B. Jiang, and Q. Tao, "Dual transfer learning for neural machine translation with marginal distribution regularization," in *Proc. AAAI*, 2018, pp. 5553–5560.

[27] B. Chen, C. Cherry, G. Foster, and S. Larkin, "Cost weighting for neural machine translation domain adaptation," in *Proc. 1st Workshop Neural Mach. Transl.*, 2017, pp. 1–7.

[28] W. Rui, M. Utiyama, L. Liu, K. Chen, and E. Sumita, "Instance weighting for neural machine translation domain adaptation," in *Proc. EMNLP*, 2017, pp. 1–7.

[29] R. Li, X. Wang, and H. Yu, "MetaMT, a meta learning method leveraging multiple domain data for low resource machine translation," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 8245–8252.

[30] B. Thompson, H. Khayrallah, A. Anastasopoulos, A. D. McCarthy, K. Duh, R. Marvin, P. McNamee, J. Gwinnup, T. Anderson, and P. Koehn, "Freezing subnetworks to analyze domain adaptation in neural machine translation," in *Proc. 3rd Conf. Mach. Transl., Res. Papers*, 2018, pp. 124–132.

[31] D. Vilar, "Learning hidden unit contribution for adapting neural machine translation models," presented at the Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol., vol. 2, 2018, doi: 10.18653/v1/N18-2080.

[32] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," 2015, *arXiv:1503.03535*.

[33] T. Domhan and F. Hieber, "Using target-side monolingual data for neural machine translation through multi-task learning," in *Proc. 2017 Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 1500–1505.

[34] D. Britz, Q. Le, and R. Pryzant, "Effective domain mixing for neural machine translation," in *Proc. 2nd Conf. Mach. Transl.*, 2017, pp. 118–126.

[35] J. Zeng, J. Su, H. Wen, Y. Liu, J. Xie, Y. Yin, and J. Zhao, "Multi-domain neural machine translation with word-level domain context discrimination," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 447–457.

[36] J. Su, J. Zeng, J. Xie, H. Wen, Y. Yin, and Y. Liu, "Exploring discriminative word-level domain contexts for multi-domain neural machine translation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 5, pp. 1530–1545, May 2021.

[37] Z.-Y. Dou, X. Wang, J. Hu, and G. Neubig, "Domain differential adaptation for neural machine translation," in *Proc. 3rd Workshop Neural Gener. Transl.*, 2019, pp. 59–69, doi: 10.18653/v1/D19-5606.

[38] M. Freitag and Y. Al-Onaizan, "Fast domain adaptation for neural machine translation," 2016, *arXiv:1612.06897*.

[39] P. Doetsch, P. Golik, and H. Ney, "A comprehensive study of batch construction strategies for recurrent neural networks in MXNet," 2017, *arXiv:1705.02414*.

[40] T. Kocmi and O. Bojar, "Curriculum learning and minibatch bucketing in neural machine translation," 2017, *arXiv:1707.09533*.

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

[42] R. Wang, M. Utiyama, and E. Sumita, "Dynamic sentence sampling for efficient training of neural machine translation," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, 2018, pp. 1–7.

[43] W. Wang, Y. Tian, J. Ngiam, Y. Yang, I. Caswell, and Z. Parekh, "Learning a multi-domain curriculum for neural machine translation," in *Proc. ACL*, 2020, pp. 1–13.

[44] E. A. Platanios, O. Stretcu, G. Neubig, B. Poczos, and T. M. Mitchell, "Competence-based curriculum learning for neural machine translation," 2019, *arXiv:1903.09848*.

[45] C. Xu, B. Hu, Y. Jiang, K. Feng, and J. Zhu, "Dynamic curriculum learning for low-resource neural machine translation," in *Proc. COLING*, 2020, pp. 1–13.

[46] X. Liu, H. Lai, D. F. Wong, and L. S. Chao, "Norm-based curriculum learning for neural machine translation," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 1–10.

[47] Y. Zhou, B. Yang, D. F. Wong, Y. Wan, and L. S. Chao, "Uncertainty-aware curriculum learning for neural machine translation," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 6934–6944.

[48] G. Kumar, G. Foster, C. Cherry, and M. Krikun, "Reinforcement learning based curriculum optimization for neural machine translation," in *Proc. NAACL*, 2019, pp. 1–7.

[49] M. Zhao, H. Wu, D. Niu, and X. Wang, "Reinforced curriculum learning on pre-trained neural machine translation models," in *Proc. AAAI*, 2020, pp. 1–8.

[50] Y. Wan, B. Yang, D. F. Wong, Y. Zhou, L. S. Chao, H. Zhang, and B. Chen, "Self-paced learning for neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 1–7.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS*, 2017, pp. 1–11.

[52] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2016, *arXiv:1409.0473v7*.

[53] J. Zeng, Y. Liu, J. Su, Y. Ge, Y. Lu, Y. Yin, and J. Luo, "Iterative dual domain adaptation for neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 1–11.

[54] A. Authors, "Multilingual neural machine translation with knowledge distillation," 2019, *arXiv:1902.10461*.

[55] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, and M. Sugiyama, "Co-teaching: Robust training of deep neural networks with extremely noisy labels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1–11.

[56] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, 1992.

[57] M. Zhao, H. Wu, D. Niu, and X. Wang, "Reinforced curriculum learning on pre-trained neural machine translation models," 2020, *arXiv:2004.05757*.
[58] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1, no. 1. Cambridge, MA, USA: MIT Press, 1998.
[59] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. ICML*, Beijing, China, 2014, pp. 387–395.
[60] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," 2015, *arXiv:1508.07909*.
[61] P. Lison and J. Tiedemann, "Opensubtitles2016: Extracting large parallel corpora from movie and TV subtitles," in *Proc. LREC*, 2016, pp. 1–7.
[62] O. Bojar, R. Chatterjee, and C. Federmann, "Findings of the 2017 conference on machine translation," in *Proc. 2nd Conf. Mach. Transl.*, 2017, pp. 1–46.
[63] T. Liang, D. Wong, L. Chao, P. Quaresma, and L. Yi, "UM-corpus: A large English-Chinese parallel corpus for statistical machine translation," in *Proc. Eur. Lang. Resour. Assoc. (LREC)*, 2014, pp. 1837–1842.
[64] Z. Shangtong. (2018). *Modularized Implementation of Deep RL Algorithms in PyTorch*. [Online]. Available: https://github.com/ShangtongZhang/DeepRL
[65] K. Heafidomain, "KenLM: Faster and smaller language model queries," in *Proc. 6th Workshop Stat. Mach. Transl.*, 2011.
[66] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. ACL*, 2002, pp. 1–8.
[67] C. Chu, R. Dabre, and S. Kurohashi, "An empirical comparison of simple domain adaptation methods for neural machine translation," 2017, *arXiv:1701.03214*.
[68] C. Kobus, J. Crego, and J. Senellart, "Domain control for neural machine translation," 2016, *arXiv:1612.06140*.
[69] Y. Kim and AM. Rush, "Sequence-level knowledge distillation," in *Proc. EMNLP*, 2016, pp. 1–11.
[70] D. Cai, Y. Wang, H. Li, W. Lam, and L. Liu, "Neural machine translation with monolingual translation memory," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process.*, vol. 1, 2021, pp. 7307–7318.
[71] T. Vu and A. Moschitti, "Machine translation customization via automatic training data selection from the web," 2021, *arXiv:2102.10243v1*.
[72] M. Del, E. Korotkova, and M. Fishel, "Translation transformers rediscover inherent data domains," 2021, *arXiv:2109.07864*.
[73] P. Henrique Martins, Z. Marinho, and A. F. T. Martins, "Efficient machine translation domain adaptation," 2022, *arXiv:2204.12608*.
[74] A. C. Stickland, A. Bérard, and V. Nikoulina, "Multilingual domain adaptation for NMT: Decoupling language and domain information with adapters," in *Proc. EMNLP 6th Conf. Mach. Transl.*, 2021.
[75] Y. Cao, H.-R. Wei, B. Chen, and X. Wan, "Continual learning for neural machine translation," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2021, pp. 3964–3974.
[76] M.-Q. Pham, F. Yvon, and J. Crego, "Latent group dropout for multilingual and multidomain machine translation," in *Proc. Findings Assoc. Comput. Linguistics (NAACL)*, 2022, pp. 1–14.
[77] M. Wu, Y. Li, M. Zhang, L. Li, G. Haffari, and Q. Liu, "Uncertainty-aware balancing for multilingual and multi-domain neural machine translation training," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 7291–7305.
[78] E. Hasler, T. Domhan, J. Trenous, K. Tran, B. Byrne, and F. Hieber, "Improving the quality trade-off for neural machine translation multi-domain adaptation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 8470–8477.

YUPENG LIU received the Ph.D. degree from the Harbin Institute of Technology. He is currently a Professor with the Harbin University of Science and Technology. His research interests include natural language processing and machine translation.

LEI ZHANG is currently pursuing the graduate degree with the Computer Science and Technology School, Harbin University of Science and Technology. His research interests include natural language processing and machine translation.

YANAN ZHANG received the Ph.D. degree from the Harbin Engineering University. He is currently an Associate Professor with the Harbin University of Science and Technology. His research interests include natural language processing and machine translation.