Improving Meta-learning for Low-resource Text Classification and Generation via Memory Imitation

Yingxiu Zhao[1], **Zhiliang Tian**[1]*, **Huaxiu Yao**[2], **Yinhe Zheng**[3], **Dongkyu Lee**[1]
Yiping Song[4], **Jian Sun**[3], **Nevin L. Zhang**[1]
[1]The Hong Kong University of Science and Technology, Hong Kong SAR, China
[2]Stanford University, [3]Alibaba Group
[4]Department of Computer Science, Peking University, Beijing, China
{yzhaocx,ztianac,dleear,lzhang}@cse.ust.hk,huaxiu@cs.stanford.edu
{zhengyinhe.zyh,jian.sun}@alibaba-inc.com,songyiping@pku.edu.cn,

Abstract

Building models of natural language processing (NLP) is challenging in low-resource scenarios where only limited data are available. Optimization-based meta-learning algorithms achieve promising results in low-resource scenarios by adapting a well-generalized model initialization to handle new tasks. Nonetheless, these approaches suffer from the *memorization overfitting* issue, where the model tends to memorize the meta-training tasks while ignoring support sets when adapting to new tasks. To address this issue, we propose a memory imitation meta-learning (MemIML) method that enhances the model's reliance on support sets for task adaptation. Specifically, we introduce a task-specific memory module to store support set information and construct an imitation module to force query sets to imitate the behaviors of some representative support-set samples stored in the memory. A theoretical analysis is provided to prove the effectiveness of our method, and empirical results also demonstrate that our method outperforms competitive baselines on both text classification and generation tasks.

1 Introduction

Building natural language processing (NLP) models in low-resource scenarios is of great importance in practical applications because labeled data are scarce. Meta-learning-based methods (Thrun and Pratt, 2012) have been commonly used in such scenarios owing to their fast adaptation ability. Notable successes have been achieved by meta-learning on low-resource NLP tasks, such as multi-domain sentiment classification (Yu et al., 2018; Geng et al., 2019) and personalized dialogue generation (Madotto et al., 2019; Song et al., 2020; Zheng et al., 2020).

Among different meta-learning approaches (Hospedales et al., 2021), optimization-based approaches have been widely used in various low-resource NLP scenarios (Madotto et al., 2019; Qian and Yu, 2019; Li et al., 2020; Mi et al., 2019) because they are model-agnostic and easily applicable. Concretely, optimization-based meta-learning algorithms aim to learn a well-generalized global model initialization θ that can quickly adapt to new tasks within a few steps of gradient updates. In the meta-training process, we first train θ on a *support set* (i.e., a few training samples of a new task i) to obtain task-specific parameters θ_i'. Then, we optimize θ based on the performance of θ_i' on a *query set* (i.e., another set of samples in task i).

Despite its effectiveness, optimization-based meta-learning algorithms usually suffer from the *memorization overfitting* issue [1] (Yin et al., 2020; Rajendran et al., 2020), where the learned model tends to solve all the meta-training tasks by memorization, rather than learning how to quickly adapt from one task to another via support sets. This is acceptable for training process, but results in poor generalization on the meta-testing sets, because the memorized model does not have knowledge of those tasks and does not know how to utilize the base learner to learn new tasks. Hence, this issue hinders the model from capturing task-specific characteristics from support sets and thus prevents the model from adapting to distinct new tasks (Rajendran et al., 2020). For instance, in personalized dialogue generation, this implies that the dialog model cannot adapt to individual users based on short conversation histories and hence fails to generate personalized responses.

Several works have been proposed to tackle the memorization overfitting issue for regression and image classification tasks. Some studies try to explicitly regularize the model parameters (Yin et al.,

*Corresponding author

[1]Memorization overfitting is different from the overfitting in conventional supervised learning (Hawkins, 2004). The latter means that the model overfits to the training tasks and fails to generalize to the testing tasks.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 583 - 595

2020; Rajendran et al., 2020), but this restricts the complexity of model initialization and reduces the model capacity. Another line of research integrates samples from support sets into the corresponding query sets via data augmentation (Yao et al., 2021). However, data augmentation on textual data may result in noisy labels or distribution shifts, which impairs the model performance (Chen et al., 2021).

In this paper, we address the memorization overfitting issue by enhancing the model's dependence on support sets when learning the model initialization, which forces the model to better leverage information from support sets. As an analogy, consider a young investor who has the ability to adapt to new circumstances rapidly but little memory of learned experiences, and an old investor who is experienced but refuses to be flexible. Our idea is to make the young investor adaptive to the various situations when he assesses his benefits so that he can not only take advantage of the old one's experience but also learn from the old investor how to leverage the learned experience. In this paper, the young investor stands for a standard meta-learning algorithm (e.g., MAML), which is prone to memorization overfitting, and the old investor is a memory module we integrate into the method, carrying information of support sets.

Specifically, we propose a Memory-Imitation Meta-Learning (MemIML) method that forces query set predictions to depend on their corresponding support sets by dynamically imitating behaviors of the latter. We therefore, introduce a memory module and an imitation module to enhance such dependence. The memory module is task-specific, storing representative information of support sets. The imitation module assists in predicting samples of query sets by dynamically imitating the memory construction. In this way, the model has to access the support set by memory imitation each time it makes a prediction on a query-set sample, hence it's no longer feasible for the model to memorize all meta tasks.

The contributions of this work are:

1. A novel method MemIML is proposed to alleviate the memorization overfitting for optimization-based meta-learning algorithms. It encourages the utilization of support sets with the help of a memory module and an imitation module when adapting to new tasks.
2. Comprehensive experiments on text classification and generation tasks show that MemIML significantly outperforms competitive baselines.
3. Theoretical proofs are given to demonstrate the effectiveness of our method.

2 Related Work

Meta-Learning. Meta-Learning aims to improve the learning algorithm itself based on the previously learned experience (Thrun and Pratt, 1998; Hospedales et al., 2021). In general, there are three categories of meta-learning methods: model-based methods, (Santoro et al., 2016; Obamuyide et al., 2019) which depend on the particular model design to facilitate fast learning; metric-based methods, (Vinyals et al., 2016; Snell et al., 2017; Geng et al., 2019) which encode samples into an embedding space and classify them based on the learned distance metric; optimization-based methods (Finn et al., 2017; Mi et al., 2019) that learn a well-generalized model initialization which allows for fast adaptation to new tasks. For low-resource scenarios in NLP, optimization-based meta-learning methods achieved promising results on tasks such as personalized dialog generation (Madotto et al., 2019; Song et al., 2020; Tian et al., 2021), low-resource machine translation (Gu et al., 2018; Sharaf et al., 2020) and question answering (Yan et al., 2020), few-shot slot tagging (Wang et al., 2021), and so on.

Memorization overfitting of Meta-learning. Meta-learning algorithms suffer from memorization overfitting. Yin et al. (2020) build an information bottleneck to the model, while this approach decreases the model performance with this passive regularization. Rajendran et al. (2020) inject random noise to the ground truth of both support and query sets, while little extra knowledge is introduced to learn a good initialization. Yao et al. (2021) address overfitting issues by augmenting meta-training tasks through mixing up support and query sets. However, such augmentation for text needs to be based on the assumption of keeping the label and the data distribution unchanged, which is often not true in practice (Chen et al., 2021). Instead of regularization and data augmentation, we leverage the support sets information stored in the memory to augment the meta-learning.

External Memory for Few-shot Learning. Memory mechanism has proven to be powerful for few-shot learning (Geng et al., 2019; Santoro et al., 2016; Munkhdalai et al., 2019). Current methods

either refine representations stored in the memory (Ramalho and Garnelo, 2018) or refining parameters using the memory (Munkhdalai and Yu, 2017; Cai et al., 2018; Wang et al., 2020). In the NLP domain, some methods store encoded contextual information into a memory (Kaiser et al., 2017; Holla et al., 2020; Zheng et al., 2019). Geng et al. (2019) propose a memory induction module with a dynamic routing algorithm for few-shot text classification tasks. Munkhdalai et al. (2019) augment the model with an external memory by learning a neural memory. Wang et al. (2021) reuse learned features stored in the memory on the few-shot slot tagging.

3 Preliminaries

We first formulate model-agnostic meta-learning (MAML) (Finn et al., 2017). Specifically, denote the base model used in MAML as f_θ and assume each task $\mathcal{T}_i$ sampled from a task distribution $p(\mathcal{T})$ associates with a dataset $\mathcal{D}_i$. Each dataset $\mathcal{D}_i$ consists of a support set $\mathcal{D}_i^s = \{(X_j^s, Y_j^s)\}_{j=1}^{N^s}$ and a query set $\mathcal{D}_i^q = \{(X_j^q, Y_j^q)\}_{j=1}^{N^q}$, where X and Y denote the input and ground truth of a sample, respectively. During the meta-training stage, a task-specific (a.k.a., post-update) model $f_{\theta_i'}$ is first obtained for each task $\mathcal{T}_i$ via gradient descent over its support set $\mathcal{D}_i^s$. Then MAML updates its initialization (a.k.a., pre-update) θ according to the performance of $f_{\theta_i'}$ on the query set $\mathcal{D}_i^q$ as in Eq.1:

$$\theta^* = \min_\theta E_{\mathcal{T}_i \sim p(\mathcal{T})} \left[\mathcal{L}\left(f_{\theta_i'}\left(X_i^q\right), Y_i^q \right) \right] \quad (1)$$

$$\text{s.t. } \theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}\left(f_\theta\left(X_i^s\right), Y_i^s \right) \quad (2)$$

where α is the inner loop learning rate. During the meta-testing stage, the learned initialization θ^* is fine-tuned on the support set $\mathcal{D}_t^s$ for task $\mathcal{T}_t$, and the resulting model is evaluated on the query set $\mathcal{D}_t^q$ with the post-update parameters θ_t'.

4 Methodology

To alleviate the memorization overfitting issue in meta-learning, we propose MemIML, which includes a memory module and an imitation module on the grounds of a base model. The memory module is task-specific, recording the mapping behaviors between inputs and outputs of support sets for each task. The imitation module is shared across tasks and predicts values for each query-set sample by dynamically imitating the memory construction. The acquired support set information leveraged by the imitation module augments the model initialization learning, enhancing the dependence of the model's task adaptation on support sets. Fig. 1 shows our model architecture.

4.1 Memory Module

We design a memory module M_i for each task $\mathcal{T}_i$ and incorporate it in the MAML framework. In order to fully leverage information from support sets, we construct key-value pairs from support-set samples and store them in the memory module. The key is the sentence representation of a sample input from support sets obtained from an introduced key network. The corresponding value is constructed to store the information of the sample output (ground truth) as in Sec. 4.3: in NLG tasks, the value is the sentence embedding of the output sentence; in NLU tasks, the value is the one hot embedding of the class label (a scalar) of the sample. Our memory has two operations: *memory writing* that constructs the memory and *memory reading* that acquires information from memory. In the following, we elaborate on these contents in detail.

Key Network represents a sample with a vector. Specifically, we use a frozen pre-trained BERT model (Devlin et al., 2019) as the key network. The input of the key network is the sample input sentence $X_j^s \in \mathcal{D}_i^s$ ($X_j^q \in \mathcal{D}_i^q$), and the output is the encoded representation of the first token (i.e. [CLS] token) of the sentence. The acquired representation is regarded as the key K_j^s for X_j^s (K_j^q for X_j^q).

Memory Writing constructs the memory using the information of samples in the support set $\mathcal{D}_i^s$. For each task $\mathcal{T}_i$, the task-specific memory M_i consists of N^i memory slots (i.e. key-value pairs $\{K_l^s, V_l^s\}_{l=1}^{N_i}$). To build these memory slots, we select samples from support sets and write their information into the memory. The sample selection is according to a diversity-based selection criterion (Xie et al., 2015) to ensure the diversity and representativeness of the memory content. The detailed description of this criterion is in Appendix. D.

For each task-specific memory module M_i, we adopt the diversity score as $S(M_i)$ on the stored keys. Here, a more diverse memory gets a higher diversity score. When the memory is not full, we directly write support-set samples without selection; otherwise, we compute the diversity score of the current memory and scores after every old key-value pair is replaced with a new key-value pair. Then we replace the old pair with the new one

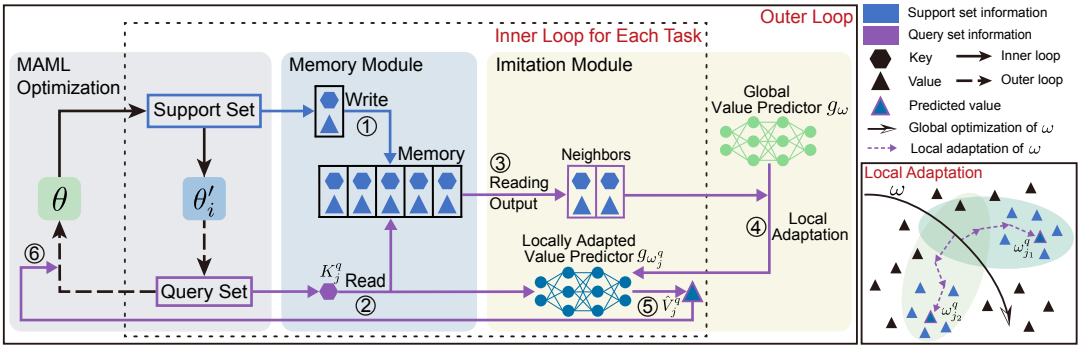


Figure 1: The architecture of our model, MemIML. The left area details the procedure of predicting a query-set sample X_j^q in each task with a task-specific memory module and an imitation module shared across tasks. The right area illustrates the local adaption of the value predictor. The two green areas represent the neighboring areas of the global parameters ω for two query-set samples in one task.

where the replacement can maximize the diversity score. In this way, the memory we build can carry more distinguishable and representative information and efficiently utilize the storage space.

Memory Reading obtains information from memory to enhance the meta-learning. The input is the sentence representation of the sample in query sets encoded by the key network, and the output is the memory slots similar to the query sample. Specifically, given the key representation K_j^q of a sample $X_j^q \in \mathcal{D}_i^q$, we retrieve the top N most similar slots from its task-specific memory M_i. The similarity is measured based on the Euclidean distance between K_j^q and each key K_l^s in the memory slots. The retrieved key-value pairs $\{K_l^s, V_l^s\}_{l=1}^N$ act as the output of memory reading.

4.2 Imitation Module

In order to better leverage the retrieved memory and enhance the dependence of our model on support sets, we propose an imitation module to encourage the imitation of support sets behaviors when making predictions on query sets. For each sample X_j^q in the query set, the inputs of the imitation module are the key K_j^q and its retrieved N memory slots, and the output is the predicted value $\hat{V}_j^q$ for X_j^q. To achieve the imitation, we construct a value predictor that can model the behaviors of support-set samples (i.e. key-value matching) stored in the memory. For estimating the value of each query-set sample, we conduct *local adaptation* on the value predictor to adapt the matching.

In this way, the proposed imitation module is customized for each query-set sample, which facilitates better capture of specific task information than directly using the memory reading output, especially when tasks are versatile. The reason is that the similarity measurement of previous memory reading operations is based on the fixed BERT representations, which ignores the task-specific information.

4.2.1 Value Predictor

In MemIML, the proposed value predictor aims to build a mapping from keys to values of the memory module mentioned in Sec. 4.1. The input of the value predictor is a key obtained from the key network, and the output is the associated value.

Specifically, we use a two-layer fully-connected network g_ω with parameters ω to build the mapping. The value predictor is learned over constructed key-value pairs of support sets across all tasks. Given the key K_j^q of a query-set sample input X_j^q, we can then estimate its associated value as $\hat{V}_j^q$.

4.2.2 Training of The Value Predictor

To train the value predictor, we minimize the reconstruction loss $\mathcal{L}_\omega^{rec}(\hat{V}, V)$ to make the predicted values as close as possible to values constructed from the ground truths of support-set samples, where $\mathcal{L}_\omega^{rec}$ is the cross-entropy loss if the value V is a label and is the mean square loss if V is a vector.

The training procedure includes the global optimization shared across tasks and the local adaptation for each specific task. Specifically, we first train the value predictor with samples from support sets of all tasks. After feeding the memory reading output of a query-set sample to this network, we perform local adaptation and employ the adapted network to estimate the value for the query sample.

Global Optimization. To obtain the task-independent global parameters ω, we train the value predictor over constructed keys (i.e., as inputs) and values (i.e., as outputs) from support-set samples of all tasks. The global optimization keeps updating in the whole meta-training phase.

Local Adaptation. To make the value predictor adaptive to each query-set sample X_j^q, inspired by (Sprechmann et al., 2018), we propose local adaptation that fine-tunes the global value predictor g_ω to get an adapted one with parameters ω_j^q. The local adaptation only works when predicting X_j^q.

Based on the initial parameters ω from the global optimization, we perform several gradient descent steps to minimize the loss $\mathcal{L}^{loc}$, which is:

$$\mathcal{L}^{loc} = \gamma \|\tilde{\omega} - \omega\|_2^2 + \frac{1}{N}\sum_{l=1}^{N} \mathcal{L}_{\tilde{\omega}}^{rec}(\hat{V}_l^s, V_l^s) \quad (3)$$

Here, $\hat{V}_l^s = g_{\tilde{\omega}}(K_l^s)$, $\{K_l^s, V_l^s\}_{l=1}^N$ is the memory reading output of the query-set sample, and the factor γ restricts the distance between ω_j^q and ω. Minimizing the second term encourages $g_{\omega_j^q}$ to better estimate the retrieved memory values $\{V_l^s\}_{l=1}^N$. Then we can acquire the locally adapted value prediction network $g_{\omega_j^q}$ with parameters $\omega_j^q = \arg\min_{\tilde{\omega}} \mathcal{L}^{loc}(\tilde{\omega})$. Given a query-sample key K_j^q, we can thus predict its associated value as

$$\hat{V}_j^q = g_{\omega_j^q}(K_j^q), \quad (4)$$

where the adapted parameters ω_j^q are discarded thereafter, and the model does not back-propagate through $\hat{V}_j^q$.

In this sense, besides the task-specific parameter θ_i' provided by MAML, there will also be ω_j^q learned from support sets specific to each query-set sample. This guarantees that the model relies more on support sets for task adaptation. Fig. 1 (right part) illustrates the mechanism of local adaptation.

4.3 MemIML on NLP Applications

In this part, we will elaborate on two few-shot applications in NLP (i.e., text generation and text classification) to solve the memorization overfitting problem of MAML. The model structures of these applications are basically the same, except for the following three points: the base model, the way to get the value V_l^s stored in the memory module, and the way to leverage the output $\hat{V}_j^q$ of Sec. 4.2.

Personalized Dialogue Generation. The base model is the transformer (Vaswani et al., 2017) consisting of an encoder and a decoder. In this task, each sample consists of an input utterance and a ground truth utterance, so the value V_l^s stored in the memory is obtained from the ground truth utterance Y_l^s of a support-set sample, which is embedded by the key network followed by an LSTM (Hochreiter and Schmidhuber, 1997). This LSTM is optimized with the base model. The $\hat{V}_j^q$, concatenated with the encoder outputs, serves as a new input for the decoder. Hence, we acquire the prediction of a query-set sample via $\hat{Y}_j^q = \text{Decoder}([\hat{V}_j^q; \text{Encoder}(X_j^q)])$.

Multi-domain Sentiment Classification. The base model is a BERT (Devlin et al., 2019) followed by a fully-connected network. Each sample consists of an input sentence and a sentiment label (ground truth), so the memory value V_l^s is the sentiment label. To leverage $\hat{V}_j^q$, we interpolate it with the original output of the base model $\tilde{Y}_j^q$ as

$$\hat{Y}_j^q = \beta \tilde{Y}_j^q + (1-\beta)\hat{V}_j^q \quad (5)$$

where β balances $\tilde{Y}_j^q$ and $\hat{V}_j^q$. Notice that the interpolation not only works on the prediction output but also guides the training via gradient descent based on the interpolated output. We verify the effectiveness of the interpolation in Appendix. C.

Algorithm 1 Memory Imitation Meta-training

Require: $p(\mathcal{T})$: task distribution, α_{1-4}: step sizes
Initialize θ from pretrained model; initialize ω randomly; initialize memory for T tasks as $\{M_i\}_{i=1}^T = \{\phi\}_{j=1}^T$
while not converge **do**
  Sample batch of tasks $\{\mathcal{T}_i\}_{i=1}^n$, where $\mathcal{T}_i \sim p(\mathcal{T})$
  for all task $\mathcal{T}_i$ **do**
    Sample support set $\mathcal{D}_i^s$ and query set $\mathcal{D}_i^q$ from $\mathcal{T}_i$
    Obtain the keys $\{K_l^s\}_{l=1}^{N^s}$ and the values $\{V_l^s\}_{l=1}^{N^s}$ for the support set $\mathcal{D}_i^s$ as in Sec. 4.1
    $M_i \leftarrow \{< K_l^s, V_l^s >\}_{l=1}^{N^s}$ # Write memory
    $\omega \leftarrow \omega - \alpha_1 \nabla_\omega \mathcal{L}^{rec}$ # Global optimization
    $\theta_i' \leftarrow \theta - \alpha_2 \nabla_\theta \mathcal{L}^{base}$ # Learn θ_i' in Eq. 2
    for (X_j^q, Y_j^q) in $\mathcal{D}_i^q$ **do**
      Obtain the keys K_j^q for each sample X_j^q
      Retrieve N nearest neighbors of K_j^q from M_i.
      $\omega_j^q \leftarrow \omega - \alpha_3 \nabla_\omega \mathcal{L}^{loc}$ # Local adaptation
      $\hat{V}_j^q = g_{\omega_j^q}(K_j^q)$ # Predict memory output
      Predict $\hat{Y}_j^q$ as in Sec. 4.3
    end for
  end for
  Update $\theta \leftarrow \theta - \alpha_4 \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i, \theta_i'}^{base}(\hat{Y}^q, Y^q)$
end while

Methods	*Automatic Metrics*												*Human Evaluation*	
	Quality							*Diversity*				*Consistency*		
	PPL	BLEU1	BLEU2	BLEU3	BLEU4	ROUGE	CIDEr	Dist1	Dist2	Dist3	Dist4	C-score	Quality	Consistency
Base Model	38.14	15.53	6.810	3.430	1.948	0.163	0.136	0.006	0.023	0.048	0.080	-0.024	0.689	0.395
Fine-tune	**34.14**	16.10	7.222	3.678	2.100	0.166	0.147	0.007	0.028	0.063	0.111	0.012	0.886	0.641
MAML	43.24	15.56	7.456	3.858	2.229	0.172	0.152	0.013	0.046	0.099	0.169	0.156	0.807	0.651
MR-MAML	52.52	13.35	5.571	2.783	1.601	0.142	0.110	0.004	0.011	0.021	0.034	0.132	0.512	0.562
MemIML	41.61	**16.23***	**7.941***	**4.295***	**2.557***	**0.183***	**0.173***	**0.014***	**0.053***	**0.114***	**0.195***	**0.241***	**0.932**	**0.807**

Table 1: Overall performance over Persona-Chat dataset. The results with * indicate that the improvements of our model overall baselines are statistically significant with $p < 0.05$ under t-test.

4.4 Theoretical Analysis

We theoretically investigate how our method helps to alleviate the memorization overfitting problem. Following Yin et al. (2020), we use mutual information $\mathcal{I}(\hat{Y}_i^q; \mathcal{D}_i^s | \theta, X_i^q)$ to measure the level of the memorization overfitting. When the learned model ignores support sets to predict query sets, $\mathcal{I}(\hat{Y}_i^q; \mathcal{D}_i^s) | \theta, X_i^q) = 0$ occurs, which indicates the complete memorization overfitting in meta-learning (Yin et al., 2020). Hence, lower mutual information means more serious memorization overfitting issues.

We propose a criterion similar to (Yao et al., 2021) to measure the validity of our method for tackling this problem. For a task $\mathcal{T}_i = \{D_i^s, D_i^q\}$, the criterion aims to mitigate the memorization overfitting by enhancing the model's dependence on the support set $\mathcal{D}_i^s$, i.e. increasing the mutual information between support set and $\hat{Y}_i^q$ as follows:

$$\mathcal{I}(\hat{Y}_i^q; [\mathcal{D}_i^s, \mathcal{M}_i] \,|\, \theta, X_i^q) > \mathcal{I}(\hat{Y}_i^q; \mathcal{D}_i^s \,|\, \theta, X_i^q), \quad (6)$$

where $\mathcal{M}_i$ means additional memory information we provide, which contains support sets information to augment the inference of the sample X_i^q in $\mathcal{D}_i^q$. We demonstrate our method MemIML meets the above criterion (See details in Appendix. A.).

4.5 The Procedure of Training and Testing

In the meta-training phase (shown in Alg. 1), MemIML first constructs an empty memory for each task and then follows the bi-level optimization process of MAML. In the inner loop, MemIML adapts the base model initialization θ to task-specific parameters via training on the support set. At the same time, from each support-set sample, MemIML obtains a key-value pair and determines whether to write it into the memory or not. Then, MemIML conducts the global optimization of the value predictor over these key-value pairs. In the outer loop, each sample of the query set reads the memory to retrieve the most similar memory slots. Local adaptation fine-tunes the value predictor on those retrieved slots. Next, the adapted value predictor estimates the value of each query sample and uses it to augment the learning of the model initialization. The total loss function in the inner loop is $\mathcal{L}^{total} = \mathcal{L}^{base} + \mathcal{L}^{rec}$, where $\mathcal{L}^{base} = \mathcal{L}(f(X^s), Y^s)$ is the cross-entropy loss.

The procedure of meta-training and meta-testing are almost the same except that meta-testing does not optimize the learned model initialization θ and the initial parameter ω of the value predictor. For each task $\mathcal{T}_t$ in the meta-testing phase, MemIML also adapts θ to task-specific parameters θ_i' in the inner-loop and constructs the task-specific memory. In the outer-loop, MemIML retrieves key-value pairs from the memory to conduct local adaptation based on the initial parameter ω. The estimated value $\hat{V}_t^q$ from local adaptation helps the base model to infer the final output $\hat{Y}_t^q$.

5 Experiments and Analysis

Experiments on personalized dialogue generation and multi-domain sentiment classification verify our model on text generation and classification, respectively, where we use Persona-Chat and ARSC datasets.

5.1 Personalized Dialogue Generation

Dataset. Following (Zhang et al., 2018), we use Persona-chat (Madotto et al., 2019) by regarding building a dialog model for each person as a task. The dataset consists of a training/validation/testing set with 1137/99/100 persons (tasks) separately. In the Persona-Chat dataset, each persona description has 8.3 unique dialogues on average, and each task consists of three samples.

Baselines. We compare our methods with the following baselines: **Base Model**: We pretrain a conventional transformer-based dialog generation

Type	Methods	Accuracy
Non meta-learning	Fine-tune	80.73
Metric-based meta-learning	Matching Net	81.22
	Prototypical Net	80.13
	Proto ++	82.41
	Relation Net	81.32
	Induction Net	79.31
Optimization-based meta-learning	MAML	82.17
	MR-MAML	78.14
	Meta-Aug	83.57
	MetaMix	83.63
	MemIML (Ours)	**85.69***

Table 2: The results of mean accuracy over the ARSC. * indicates that our improvement overall baselines is statistically significant with $p < 0.01$ under t-test.

model over all the training tasks ignoring the speakers' personality. **Fine-tune**: We fine-tune the pre-trained base model on the support sets of each meta-testing task. **MAML**: We apply MAML (Madotto et al., 2019) to the base model. **MR-MAML**: Yin et al. (2020) tackle the memorization overfitting of MAML via regularization.

Metrics. Automatic evaluation has three aspects,

- *Quality*: **BLEU-n** (Papineni et al., 2002), **CIDEr** (Vedantam et al., 2015), and **ROUGE** (Lin, 2004) measures the n-gram matching between the generated response and ground truth. **PPL** (perplexity) measures the sentence fluency.
- *Diversity.* **Dist-n** (Li et al., 2016) evaluates the response diversity by counting unique n-grams.
- *Consistency*: **C score** (Madotto et al., 2019) measures the consistency between the generated responses and persona descriptions through a pre-trained natural language inference model.

Human evaluation consists of **Quality** and **Consistency**. (See details in Appendix. B.1).

Overall Performance. As shown in Table 1. **Fine-tune** outperforms **Base Model** in all metrics, which verifies that the task-specific data is helpful to its performance on specific tasks. Compared to **Fine-tune**, **MAML** behaves better on diversity and consistency but behaves worse on quality. Pre-training the base model achieves the best perplexity (lowest PPL) as shown by **Base Model** and **Fine-tune**. We analyze that it's because pretraining leads to a considerable degree of fluency in their generated utterances and is careless about each task's specific information, resulting in low consistency with tasks. Our model, **MemIML**, performs the best in most aspects, including quality, diversity, and task consistency. In particular, MemIML significantly improves MR-MAML in alleviating the memorization overfitting issue, suggesting that memory imitation is more effective than only regularizing model initialization.

5.2 Multi-domain Sentiment Classification

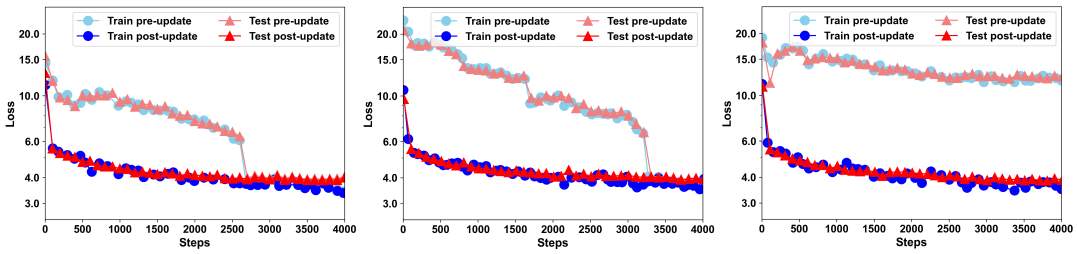
Dataset. Amazon Review sentiment classification dataset (ARSC) (Yu et al., 2018) contains 69 tasks in total. Following (Geng et al., 2019), we build a 2-way 5-shot meta-learning with 57 tasks for meta-training and 12 tasks for meta-testing. We conduct experiments on the ARSC (Yu et al., 2018). It contains English reviews of 23 types of Amazon products, where each product consists of three different binary classification tasks. Following Geng et al. (2019), we select 12 tasks from 4 domains (*Books, DVD, Electronics, Kitchen*) for meta-testing tasks, and the support sets of these tasks are fixed (Yu et al., 2018).

Baselines. We compare our methods with the following baselines: **Fine-tune**: We fine-tune a pre-trained BERT on the support set of meta-testing tasks (non-meta-learning method) as in Appendix. B.2. We choose five metric-based meta-learning baselines: **Matching Net** (Vinyals et al., 2016), **Prototypical Net** (Snell et al., 2017), **Proto ++**, (Ren et al., 2018), **Relation Net** (Sung et al., 2018), and **Induction Net** (Geng et al., 2019). We apply an optimization-based baseline (**MAML**) (Finn et al., 2017) to the base model, and implement some approaches tackling the memorization overfitting problem based on MAML: **MR-MAML** (Yin et al., 2020), **MetaMix**, (Yao et al., 2021) and **Meta-Aug** (Rajendran et al., 2020).

Overall Performance. Table 2 shows the performance measured by the mean accuracy of meta-testing tasks. Our model, **MemIML** outperforms all competing approaches including non-meta-learning, metric-based meta-learning, and optimization-based meta-learning methods. Particularly, our model surpasses the current solutions to the memorization overfitting problem (**MR-MAML, Meta-Aug, MetaMix**), indicating that our method is more effective compared to regularization and textual augmentation.

5.3 Memorization Overfitting Analysis

In Figure 2, the gaps of the losses on query sets between pre-update θ (before training on support sets)



(a) MAML (b) MR-MAML (c) MemIML (Ours)

Figure 2: Memorization overfitting analysis on Persona-Chat. Small loss gaps between pre-update θ and post-update θ'_i (in MAML and MR-MAML) indicate the serious memorization overfitting issue (i.e., the gap between sky-blue and blue curves in meta-training and the gap between pink and red curves in meta-testing). The large gap in MemIML demonstrates the effectiveness of our method.

	Persona-Chat								ARSC
	PPL	C-score	BLEU3	BLEU4	Dist1	Dist2	ROUGE	CIDEr	Acc
MemIML	**41.62**	**0.240**	**4.295**	**2.557**	**0.014**	**0.053**	**0.183**	**0.173**	**85.69**
- Similarity-Search	45.17	0.153	3.817	2.219	0.011	0.044	0.168	0.158	84.14
- Value predictor	42.93	0.183	4.199	2.313	0.010	0.039	0.182	0.167	84.67
- Local Adaptation	48.08	-0.117	3.452	1.948	0.007	0.023	0.171	0.129	84.19

Table 3: Ablation Studies. - means deleting MemIML's components.

Memory Analysis on ARSC			
Store ratio	Acc	# Neighbors	Acc
100%	84.91	5	84.04
80%	85.69	10	84.47
50%	84.84	20	85.69
20%	84.35	50	85.04

Table 4: Memory analysis on ARSC.

and post-update θ'_i (after training on support sets) indicate the memorization overfitting problem. The gap between sky-blue and blue curves measures the memorization overfitting of meta-training (the gap between pink and red curves measures meta-testing). Small loss gaps indicate a severe memorization overfitting where support sets are almost useless for task adaptation. Those loss gaps between θ and θ'_i collapse in MAML and MR-MAML after about 3000 steps. This indicates that the post-update θ'_i barely benefits from the support set, and thus the memorization overfitting issue is severe. In Figure 2 (c), MemIML has large gaps between θ and θ'_i, implying that θ'_i better leverages support sets when adapting to new tasks and thus alleviates the memorization overfitting issue.

5.4 Ablation Studies

In Table 3, we conduct ablation studies to verify the effectiveness of each component. Removing *Similarity-Search* means the memory reading operation randomly outputs memory slots instead of searching for similar memory slots. This variant underperforms MemIML, indicating that similar samples stored in the memory provide more useful information to improve the model performance. Removing the *value predictor* means directly using the memory output without a learnable network. Its results are not too bad, indicating that the memory module helps to mitigate the memorization overfitting problem. However, this usage simply aggregates the support set information into the query set, which is not as precise as learning the information required by the query set itself. Therefore, it is still inferior to our model. Removing *Local adaptation* means we only use the global value predictor to estimate the memory output. It is crucial to the value predictor since removing it from the value predictor results in an even worse performance than removing the *value predictor*. Besides, the significant drop in task consistency (C-score) shows that local adaptation contributes a lot to making the model adaptive to specific tasks, as it learns to adapt to each query-set sample.

5.5 Analysis of Memory Operations

Memory Size. In Table 4 and 5, we investigate the variants of our task-specific memory module of different sizes. We control the memory size through $|M| = \text{store ratio} \times |D^s|$. The results demonstrate that our model is able to maintain high performance even with only a 20% memory size by storing diverse and representative samples of support sets. Besides, as the ratio of stored samples increases, the model's performance is improved since it provides more information for the inference of query samples and the optimization of the model initialization. Storing all the encountered samples (i.e., with store ratio 100%) in the memory instead introduces some noise that damages the model performance.

		PPL	C-score	BLEU3	BLEU4	Dist1	Dist2	ROUGE	CIDEr
Store ratio	1	43.54	0.197	4.224	2.447	0.014	0.055	0.179	0.174
	0.8	43.21	0.198	4.414	2.622	0.014	0.054	0.182	0.183
	0.5	41.86	0.223	4.069	2.317	0.013	0.052	0.179	0.162
	0.2	41.97	0.204	4.021	2.271	0.012	0.052	0.181	0.168
Neighbor number	5	41.98	0.192	3.855	2.203	0.013	0.053	0.177	0.162
	10	41.62	0.239	4.295	2.557	0.014	0.053	0.183	0.173
	20	42.12	0.155	4.099	2.336	0.012	0.046	0.179	0.165
	50	42.76	0.145	3.614	2.072	0.011	0.041	0.169	0.144

Table 5: Analysis of Persona-Chat dataset.

Number of Neighbors. We also investigate the effects of different numbers of neighbors for the model performance in Table 4 and Table 5. In both datasets, the model performs better with a larger number of neighbors. However, when the number of neighbors is too large, the model retrieves some dissimilar slots from the memory module. These dissimilar slots bring much noise, which makes the predictions of query samples inaccurate.

5.6 Case Study

We present two generated cases in personalized dialog in Table. 6. **Base Model**, **Fine-tune**, and **MAML** generate general responses with little useful information or responses that are not consistent with the personality of personas. **MR-MAML** generates irrelevant responses to the dialogue context. Our model not only responds coherently to the dialog history but also caters to the persona descriptions of each user.

Persona A	I am a professional singer. I graduated from USC. I love watching horror films. I was born in South Carolina.
Dialogues	A: Where are you from? I grew up in South Carolina. B: I was born in Arkansas, but I now live in Chicago. A: Did you go to college there? B: Yes, I am in school now while working part-time as a barista.
Human Response	I graduated recently from USC.My parents pushed me into it.
Base Model	What do you do for a living?
Fine-tune	Are you in trouble?
MAML	Do you work in the fitness industry?
MR-MAML	That sounds illegal. Are they not protected?
MemIML	That would be cool! I love horror films.
Persona A	I have a large garden. My hobby is walking. I love to travel. I teach biology at a college.
Dialogues	B: Hi, Merna here. Single, 32 and scared of dogs. You? A: I am dave, 40. B: Oh, okay. I am a good cook!
Human Response	Cool, I grow my own vegetables in my garden garden.
Base Model	What do you do for a living?
Fine-tune	What do you teach?
MAML	What do you do?
MR-MAML	I am doing great. How are you tonight?
MemIML	Oh, i am a science teacher and love travel.

Table 6: Two generated examples in the Persona-Chat dataset. Colored texts indicate that the responses match the personality descriptions.

6 Conclusion

In this paper, we tackle the memorization overfitting problem of meta-learning for text classification and generation applications. We propose MemIML to enhance the dependence of the model on the support sets for task adaptation. MemIML introduces a memory module storing the information of support sets, and propose an imitation module to better leverage the support set information by imitating the behaviors of the memory. Both empirical and theoretical results demonstrate that our method MemIML effectively alleviates the memorization overfitting problem.

7 Ethical Considerations

The persona-based dialogue generation task aims to build a dialogue model which generates meaningful, fluent, and consistent responses. It will facilitate human-computer interactions in practice. However, the training of the model for personalized dialogues may lead to the leakage of personal privacy information. In this work, the data source we use is from a published dataset and does not involve privacy issues for the data collection. Our proposed method does not include inference or judgments about individuals and does not generate any discriminatory, insulting responses. Our work validates the proposed method and baseline models on human evaluation which involves manual labor. We hire five annotators to score 750 generated sentences in total (250 sentences for each model we evaluate). The hourly pay is set to 15 US$ per person, which is higher than the local statutory minimum wage.

Acknowledgements

Research on this paper was supported by Hong Kong Research Grants Council (Grant No. 16204920) and National Natural Science Foundation of China (Grant No. 62106275).

References

Qi Cai, Yingwei Pan, Ting Yao, Chenggang Yan, and Tao Mei. 2018. Memory matching networks for one-shot image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4080–4088.

Jiaao Chen, Derek Tam, Colin Raffel, Mohit Bansal, and Diyi Yang. 2021. An empirical survey of data augmentation for limited data learning in nlp. *arXiv e-prints*, pages arXiv–2106.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2018. Universal transformers. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

Thomas Dopierre, Christophe Gravier, and Wilfried Logerais. 2021. A neural few-shot text classification reality check. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 935–943.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR.

Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. Induction networks for few-shot text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3904–3913.

Jiatao Gu, Yong Wang, Yun Chen, Victor OK Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631.

Douglas M Hawkins. 2004. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Nithin Holla, Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. 2020. Learning to learn to disambiguate: Meta-learning for few-shot word sense disambiguation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4517–4533.

Timothy M Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. 2021. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Rumeng Li, Xun Wang, and Hong Yu. 2020. Metamt, a meta learning method leveraging multiple domain data for low resource machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8245–8252.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Andrea Madotto, Zhaojiang Lin, Chien-Sheng Wu, and Pascale Fung. 2019. Personalizing dialogue agents via meta-learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5459.

Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. 2019. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. *arXiv preprint arXiv:1905.05644*.

Tsendsuren Munkhdalai, Alessandro Sordoni, TONG WANG, and Adam Trischler. 2019. Metalearned neural memory. *Advances in Neural Information Processing Systems*, 32:13331–13342.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563. PMLR.

Abiola Obamuyide, Andreas Vlachos, et al. 2019. Meta-learning improves lifelong relation extraction.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Kun Qian and Zhou Yu. 2019. Domain adaptive dialog generation via meta learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2639–2649.

Janarthanan Rajendran, Alexander Irpan, and Eric Jang. 2020. Meta-learning requires meta-augmentation. *Advances in Neural Information Processing Systems*, 33:5705–5715.

Tiago Ramalho and Marta Garnelo. 2018. Adaptive posterior learning: few-shot learning with a surprise-based memory module. In *International Conference on Learning Representations*.

Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. 2018. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR.

Amr Sharaf, Hany Hassan, and Hal Daumé III. 2020. Meta-learning for few-shot nmt adaptation. In *NGT@ ACL*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4080–4090.

Yiping Song, Zequn Liu, Wei Bi, Rui Yan, and Ming Zhang. 2020. Learning to customize model structures for few-shot dialogue generation tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5832–5841.

Pablo Sprechmann, Siddhant M Jayakumar, Jack W Rae, Alexander Pritzel, Adria Puigdomenech Badia, Benigno Uria, Oriol Vinyals, Demis Hassabis, Razvan Pascanu, and Charles Blundell. 2018. Memory-based parameter adaptation. In *International Conference on Learning Representations*.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208.

S. Thrun and L. Pratt. 1998. *Learning to Learn: Introduction and Overview*. Learning to Learn: Introduction and Overview.

Sebastian Thrun and Lorien Pratt. 2012. *Learning to learn*. Springer Science & Business Media.

Zhiliang Tian, Wei Bi, Zihan Zhang, Dongkyu Lee, Yiping Song, and Nevin L Zhang. 2021. Learning from my friends: Few-shot personalized conversation systems via social networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13907–13915.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575. IEEE Computer Society.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638.

Hongru Wang, Zezhong Wang, Gabriel Pui Cheong Fung, and Kam-Fai Wong. 2021. Mcml: A novel memory-based contrastive meta-learning method for few shot slot tagging. *arXiv preprint arXiv:2108.11635*.

Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM Comput. Surv.*, 53(3).

Pengtao Xie, Yuntian Deng, and Eric Xing. 2015. Diversifying restricted boltzmann machine for document modeling. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1315–1324.

Ming Yan, Hao Zhang, Di Jin, and Joey Tianyi Zhou. 2020. Multi-source meta transfer for low resource multiple-choice question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7331–7341, Online. Association for Computational Linguistics.

Huaxiu Yao, Long-Kai Huang, Linjun Zhang, Ying Wei, Li Tian, James Zou, Junzhou Huang, et al. 2021. Improving generalization in meta-learning via task augmentation. In *International Conference on Machine Learning*, pages 11887–11897. PMLR.

Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. 2020. Meta-learning without memorization. In *International Conference on Learning Representations*.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. In *Proceedings of*

the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1206–1215.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213.

Yinhe Zheng, Guanyi Chen, Minlie Huang, Song Liu, and Xuan Zhu. 2019. Personalized dialogue generation with diversified traits. *arXiv preprint arXiv:1901.09672*.

Yinhe Zheng, Rongsheng Zhang, Minlie Huang, and Xiaoxi Mao. 2020. A pre-training based personalized dialogue generation model with persona-sparse data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9693–9700.

A Validity of Memory Imitation Strategy

Proof of inequality in Eqn. 6. We check the validity of memory imitation by examining whether the criterion in Section 4.4 is met. We check the increase of mutual information between predictions of query sets with the provided support-set information after augmented with the memory information $\mathcal{M}$.

$$\begin{aligned}
&\mathcal{I}(\hat{Y}^q; [D^s, \mathcal{M}]|\theta, X^q) - \mathcal{I}(\hat{Y}^q; D^s|\theta, X^q) \\
&= H(\hat{Y}^q|\theta, X^q) - H(\hat{Y}^q|D^s, \mathcal{M}, \theta, X^q) \\
&\quad - H(\hat{Y}^q|\theta, X^q) + H(\hat{Y}^q|D^s, \theta, X^q) \\
&= -H(\hat{Y}^q|X^q, X^s, Y^s, \mathcal{M}, \theta) \\
&\quad + H(\hat{Y}^q|X^q, X^s, Y^s, \theta). \qquad (7)
\end{aligned}$$

For short, we use notation $\boldsymbol{Z} = (X^q, X^s, Y^s, \theta)$ to denote a set of variables. Then we can rewrite (7) as

$$\begin{aligned}
&- H(\hat{Y}^q|\boldsymbol{Z}, \mathcal{M}) + H(\hat{Y}^q|\boldsymbol{Z}) \\
&= E_{\hat{Y}^q, \boldsymbol{Z}, \mathcal{M}} \left[\log p(\hat{Y}^q|\boldsymbol{Z}, \mathcal{M})\right] \\
&\quad - E_{\hat{Y}^q, \boldsymbol{Z}} \left[\log p(\hat{Y}^q|\boldsymbol{Z})\right].
\end{aligned}$$

Note that trivially, we have $E_{\mathcal{M}}[1] = 1$, so we get

$$E_{\hat{Y}^q, \boldsymbol{Z}} \left[p(\hat{Y}^q|\boldsymbol{Z})\right] = E_{\hat{Y}^q, \boldsymbol{Z}, \mathcal{M}} \left[p(\hat{Y}^q|\boldsymbol{Z})\right]$$

since $p(\hat{Y}^q, \boldsymbol{Z})$ does not rely on the variable $\mathcal{M}$. Hence, we can just write $E_{\hat{Y}^q, \boldsymbol{Z}, \mathcal{M}}$ as E for short. Then the equation (7) will become to

$$\begin{aligned}
&E[\log p(\hat{Y}^q|\boldsymbol{Z}, \mathcal{M})] - E[\log p(\hat{Y}^q|\boldsymbol{Z})] \\
&= E[\log \frac{p(\hat{Y}^q|\mathcal{M}, \boldsymbol{Z})}{p(\hat{Y}^q|\boldsymbol{Z})}] \\
&= \sum_{\hat{Y}^q, \mathcal{M}, \boldsymbol{Z}} p(\boldsymbol{Z}) p(\hat{Y}^q, \mathcal{M}|\boldsymbol{Z}) \log \frac{p(\hat{Y}^q, \mathcal{M}|\boldsymbol{Z})}{p(\hat{Y}^q|\boldsymbol{Z}) p(\mathcal{M}|\boldsymbol{Z})} \\
&= E_{\boldsymbol{Z}}[KL(p(\mathcal{M}, \hat{Y}^q|\boldsymbol{Z}) \| p(\hat{Y}^q|\boldsymbol{Z}) p(\mathcal{M}|\boldsymbol{Z}))] \\
&> 0
\end{aligned}$$

where the last inequality holds due to $\hat{Y}^q$ is dependent on $\mathcal{M}$. □

We also investigate that memory imitation improves the learning of model initialization via another criterion $\mathcal{I}(\theta; [D^q, \mathcal{M}]|D^q) > 0$ following Yao et al. (2021). This criterion guarantees that the additional memory knowledge contributes to updating the initialization in the outer loop. Since

all the meta-training tasks satisfy this criterion, the generalization ability of the model initialization improves.

Proof.

$$\begin{aligned}
&\mathcal{I}(\theta; [D^q, \mathcal{M}]|D^q) \\
&= H(\theta|D^q) - H(\theta|D^q, \mathcal{M}) \\
&= E[-\log P(\theta|D^q)] + E[\log p([\theta|D^q, \mathcal{M}])] \\
&= E[\log \frac{p(\theta|D^q, \mathcal{M})}{p(\theta|D^q)}] > 0
\end{aligned}$$

□

B Experimental Details

B.1 Personalized Dialogue Generation

Experimental Setup. We implement our model based on the transformer (Dehghani et al., 2018; Vaswani et al., 2017) with pre-trained Glove embedding (Pennington et al., 2014) following (Madotto et al., 2019). The hidden dimensions of the LSTM unit are set to 1024. We set the number of neighbors $N = 10$ and the number of local adaptation steps $L = 20$. We follow all other hyperparameter settings in Madotto et al. (2019): we use SGD for the inner loop training and Adam for the outer loop update with learning rates 0.01 and 0.0003, respectively. We set batch size as 16 and use beam search with beam size 5.

Human Evaluation We conduct human evaluation following Song et al. (2020) considering two aspects **Quality** and **Consistency** where five well-educated volunteers annotate 250 generated responses for each model. The annotators score each response from two aspects: **Quality** and **Consistency** in a 3-point scale: 2 for good, 1 for fair, and 0 for bad. *Quality* measures coherence, fluency, and informativeness. *Consistency* measures the task consistency between the generated responses and the person's persona description.

B.2 Multi-domain Sentiment Classification

Experimental Setup. We utilize a BERT (Devlin et al., 2019) as the encoder. We fine-tune the off-the-shelf pre-trained BERT on the masked language modeling task following (Dopierre et al., 2021) as it greatly improves embeddings' quality (Sun et al., 2019). The fine-tuned BERT is then used as the initialization for all few-shot models. We use Adam (Kingma and Ba, 2015) optimizer for both inner and outer loop update with learning rate $2e^{-5}$ and $1e^{-5}$ respectively, and we set $\beta = 0.2$ in Eqn. 5, the number of neighbors $N = 20$ and the number of local adaptation steps $L = 5$.

C Effectiveness of the Interpolation

To measure whether MemIML improves the learned model initialization, we add an experiment that does not incorporate the memory module during meta-testing (i.e., $\beta = 1$ in Eq. 5) for the multi-domain sentiment classification task. The better result of MemIML than MAML and other regularization methods demonstrate the superiority of our model.

Model	Mean Accuracy
MAML	82.17
MR-MAML	78.14
Meta-Aug	83.57
MetaMix	83.63
MemIML ($\beta = 1$)	84.95

Table 7: Comparison of mean accuracy on the ARSC.

D Diversity-selection Criterion

For each task-specific memory module M, following Xie et al. (2015), we adopt the diversity score as $S(M) = \mu(M) - \sigma(M)$ on the stored keys, where $\mu(M) = \frac{1}{N^2}\sum_{j=1}^{N}\sum_{h=1}^{N}\angle(K_j, K_h)$ denotes the mean of angles between every two stored key representations and $\sigma(M) = \frac{1}{N^2}\sum_{j=1}^{N}\sum_{h=1}^{N}(\angle(K_j, K_h) - \mu(M))^2$ denotes the variance of those angles [2].

[2] $\angle(K_j, K_h) = arccos(\frac{K_j \cdot K_h}{\|K_j\|_2 \|K_h\|_2})$