Meta-learning with Prototypical Query Sample Editing for Low Resource Continual Learning

Anonymous ACL submission

Abstract

Memory replay, also known as experience rehearsal, is a mainstream method to retain knowledge in continual learning (CL), especially in NLP. Recently, meta-learning has been introduced to augment memory replay. It serves to enable efficient knowledge transfer, thereby alleviating catastrophic forgetting. However, memory replay is often episodic to avoid over-fitting to past samples. Its sparse occurrence also limits the convergence of model on past samples, especially in low resource scenarios. This paper aims to fully exploit the potential of meta-learning. We study the feasibility of solely using meta-learning to solve lifelong language learning. We propose an optimization-based meta-learning framework for CL in accordance with MAML. In particular, we edit query information for meta-learning via a prototypical network. The meta-objective is modified to depict a CL scenario. We conduct extensive experiments on benchmark text classification datasets. The results testify the superiority of our method in terms of forgetting mitigation, fast adaptation and memory efficiency in a low resource NLP scenario.

1 Introduction

Existing continual learning (CL) algorithms update model parameters whenever a new task arrives. The important information from earlier tasks can be easily erased or overwritten when the data distribution shifts. Consequently, catastrophic forgetting (McCloskey and Cohen, 1989) occurs and harms performance on preceding tasks. Especially for language models, the number of parameters in a language model is usually huge. A small change in parameter space could impact model output unexpectedly (Wang et al., 2019). Hence, replaying past examples is the mainstream approach for lifelong language learning. However, existing replay-based methods not only easily over-fit to past samples (Zhao et al., 2022), but also cause under-fitting problem on low resource tasks.

Meta-learning can enable efficient knowledge transfer. Recently, it has been introduced as a learning framework for CL. Mostly, meta-learning serves to perform efficient knowledge transfer via fast adaptation and facilitate memory replay. Memory replay is often episodic to avoid over-fitting problem. This episodic nature is unfavourable for cross-domain fast adaptation when training examples are insufficient and memory budget for past samples is small.

To address this dilemma, we propose a method to solve low resource CL problems, namely MAML-CL. Specifically, we use Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017) as our meta-learning framework. We argue that meta-learning can solve CL problems without episodic memory replay. The mainstream CL methods limit the change of non-trivial weights or parameters, hereby retaining knowledge. In meta-learning, the variation on parameter space is restricted by meta-objective. We use a prototypical network to select representative past examples for query set. It modifies the meta-objective and controls the learning process on the new task, thereby ameliorating catastrophic forgetting.

We conduct extensive experiments on CL benchmark datasets from (Zhang et al., 2015), popularized by (de Masson d'Autume et al., 2019) in lifelong language learning. This collection of datasets includes text classification datasets from diverse domains. We reduce the size of the training set for each task to its 10%, i.e., 11,500 per task. We demonstrate our method's robustness to forgetting mitigation, fast adaptation and memory efficiency in a low-resource scenario.

The contributions of this work are threefold: (1) Our method validates the feasibility of using meta-learning without sparse experience replay, the mainstream lifelong language learning approach; (2) We design a prototypical query example editing method to adapt the meta-objective to a CL

scenario. Our method addresses low resource CL problems. It well-preserves fast adaptation ability of meta-learning while avoiding catastrophic forgetting; (3) We perform extensive experiments in a low resource scenario. The results testify the superiority of our method on forgetting mitigation, fast adaptation and memory efficiency.

2 Related Work

Robust continual learning aims to guarantee the *stability* of handling various tasks, while showing *plasticity* on a novel domain. Existing approaches can be categorised into two main mainstreams, i.e., regularization-based methods (Kirkpatrick et al., 2016; Li and Hoiem, 2016; Zenke et al., 2017) and replay-based methods (Wang et al., 2019, 2020; Chaudhry et al., 2019; Sun et al., 2019). In general, regularization-based methods retain the updates of non-trivial parameters or weights by adding constraints or penalty. Generally, language models are deep learning networks. Given the complexity of deep learning models, memory replay-based approaches are broadly considered as a plausible approach for lifelong language learning.

Replay-based methods save a certain amount of previous training data and retrieval them while training for a novel domain. Many models used random selection strategy to select or save past examples. But it always failed to consider sample efficiency. Recent replay-based methods utilised episodic memory replay with a 1% replay rate. It was first proposed in MbPA++ (de Masson d'Autume et al., 2019). MbPA++ retrieved K nearest neighbours via Euclidean distance function for local adaptation, which is expensive in its inference process. To address this problem, (Holla et al., 2020) introduced a replay frequency to harness the replay sparseness in terms of time and size. Nevertheless, they neglected memory constraints. The size of occupied memory is explosively expanding with the increase of training samples.

Meta-learning in CL. Recently, meta-learning has been introduced into CL models, considering its ability of fast adaptation and knowledge transfer. Recent works employed MAML (Finn et al., 2017) to improve initial parameters. It can fast adapt to various domains with few learning samples. Meta-MbPA (Wang et al., 2020) performed local adaptation with episodic memory and used MAML to find a better initialized state for local adaptation. OML-ER (Holla et al., 2020) and ANML-ER (Holla et al., 2020) utilised an online meta-learning model and a neuromodulated meta-learning respectively for fast adaptation, augmented with sparse experience replay. Additionally, some CL models used Reptile (Nichol et al., 2018) as their meta-learning algorithms. MER (Riemer et al., 2019) regularized the objective of experience replay via a modified Reptile (Nichol et al., 2018) algorithm and memory replay module. MLLRE (Obamuyide and Vlachos, 2019) also adopted Reptile to meta updates parameters via augmented training set.

3 Problem Formulation

3.1 Continual Learning

In a basic CL setup, the learner ingests a stream of training examples. We assume the training stream consists of K tasks, $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_K\}$, in an ordered sequence. Given the latest task $\mathcal{T}_K$ and a CL learner f_{θ_K}, the objectives are: (a) to finetune parameter set θ_K over a parameter space Θ, such that the task objective $\mathcal{L}_{\mathcal{T}_K}$ is minimal. That is,

$$\tilde{\theta}_K = \arg\min_{\theta_K \in \Theta} \mathcal{L}_{\mathcal{T}_K}(\theta_K) \tag{1}$$

(b) to perform well with the learned parameter set $\tilde{\theta}_K$ on all preceding tasks $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_{K-1}\}$ without using the past training sets. Assuming that all tasks are equally important, the overall objective is to minimise the expected risk of all seen tasks, $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_K\}$, as:

$$\min_{\tilde{\theta}_K} \sum_{k=1}^{K} \mathbb{E}_{\mathcal{T}_k}[\mathcal{L}_{\mathcal{T}_k}(\tilde{\theta}_K)] \tag{2}$$

A replay-based method allows the model to retain a certain amount of past instances. Typically, the basic CL setup allows the memory buffer size to a constant B, where the amount of past training data in memory should not exceed B.

In this paper, we consider a popular scenario of continual learning, i.e., *class-incremental learning* (CIL), where task identity information is not provided in inference, and a single classifier is for all classes. We leverage cross entropy loss $\mathcal{L}_{CE}$ as the loss of a task $\mathcal{T}_k$ as,

$$\begin{aligned} \mathcal{L}_{\mathcal{T}_k}(\theta) &= \mathcal{L}_{CE}(\boldsymbol{x}, \boldsymbol{y}; \theta) \\ &= -\sum_{j=1}^{|D_k|} \sum_{i=1}^{N_k} y_{ji} \log(\sigma(f_\theta(x_j))_i) \end{aligned} \tag{3}$$

where k is the task index, $D_k \subseteq D^k_{\text{train}}$, D^k_{train} is the ground truth label set of $\mathcal{T}_k$, N_k is the number

of classes in $\mathcal{T}_k$ and σ is the activation function (e.g., sigmoid or softmax).

3.2 Meta-learning

Model agnostic meta-learning (MAML) (Finn et al., 2017) algorithm is known as "think in advance" by learning an optimal initial state of an algorithm. We assume a task $\mathcal{T}$ and a set of initial parameters ϕ over a parameters space Φ. We expect ϕ to yield a minimal loss after m updates in $\mathcal{T}$ as,

$$\min_{\phi} \mathbb{E}_{\mathcal{T}}[\mathcal{L}_{\mathcal{T}}(U^m_{\mathcal{T}}(\phi))] \tag{4}$$

where $U^m_{\mathcal{T}}$ is the update operation that performs m times gradient-based updates on parameters ϕ, using samples drawn from distribution $p(\mathcal{T})$. In an online MAML setting, each episode uses m batches of training instances as a support set S. The inner loop algorithm performs m-step SGD on parameters ϕ in the inner loop, as:

$$\begin{aligned}\phi^* &= U^m_{\mathcal{T}}(\phi) \\ &= U_{\mathcal{T},S}(\phi) \\ &= \phi - \alpha \nabla_{\phi} \mathcal{L}_{\mathcal{T}}(S, \phi)\end{aligned} \tag{5}$$

where α is the stepsize. Test examples that specified problems in outer loop algorithm are a set of query samples Q. The updated ϕ^* are further optimized on Q to achieve meta objective as,

$$\min_{\phi} \mathbb{E}_{\mathcal{T}}[\mathcal{L}_{\mathcal{T},Q}(\phi^*)] = \min_{\phi} \mathbb{E}_{\mathcal{T}}[\mathcal{L}_{\mathcal{T},Q}(U_{\mathcal{T},S}(\phi))] \tag{6}$$

4 Prototypical Query Sample Editing

To align meta-objective with CL objective, we edit query set Q to generalize all seen tasks, $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_K\}$. Particularly, we select representative past examples as query examples for meta-learning. We apply a prototypical network for query sample selection, where each class is characterized by a prototype.

4.1 Dynamic Prototype Computation

We devise a prototypical network, $h_{\phi_{\text{proto}}}$, for prototype computation. Each prototype $\mathbf{c}_l$ is computed as the mean vector of the embedded support examples from the same class l, as:

$$\mathbf{c}_l \leftarrow \frac{1}{|S_l|} \sum_{(x_j, y_j) \in S_l} h_{\boldsymbol{\phi}_{proto}}(x_j) \tag{7}$$

where l is the class index and S_l is the support set for class l. The devised prototypical network $h_{\phi_{\text{proto}}}$ is meta-learnable and dynamically compute and update prototypes in each episode. And, we include the prototypical loss J_P in inner loop loss $\mathcal{L}_{\text{inner}}$ as a regularization term. For a task $\mathcal{T}_k$, the prototypical loss J_P (Snell et al., 2017) is,

$$\begin{aligned}J_P(\phi_{\text{proto}}) = \frac{1}{N_k} \sum_{l=1}^{N_k} \Big(\frac{1}{|Q_l|} \big[d(h_{\phi_{proto}}(\boldsymbol{x}_l), \mathbf{c}_l) \\ + \log \sum_{l'} \exp(-d(h_{\phi_{proto}}(\boldsymbol{x}_l), \mathbf{c}'_l)) \big] \Big)\end{aligned} \tag{8}$$

where Q_l is the query set for class l, $(\boldsymbol{x}_l, y_l) \in Q_l$, N_k is the number of classes in $\mathcal{T}_k$ and $d(\cdot)$ is a distance function. In this paper, we leverage Euclidean distance.

Algorithm 1: Meta-training

Input: Initial parameters $\theta = \phi_{\text{proto}} \cup \phi_{\text{pred}}$, training set D_{train}, support set size m, episode index i, memory buffer D_{Memory}, No. of support examples for each class N_S, No. of query examples for each class N_Q, inner-loop learning rate α, outer-loop learning rate β, and No. of selected samples per class N_{select}.
Output: Trained parameters θ and memory D_{Memory}

for $i = 1, 2, ...$ **do**
  $S \leftarrow m$ batches of examples D^i_{train} from the stream
  [Prototype Computation]
  $J_p(\phi_{\text{proto}}) \leftarrow 0$
  for class l in S **do**
    Select N_S examples of class l from S as S_l.
    Select N_Q examples of class l from $S \backslash S_l$ as Q_l.
    Compute and update prototype $\boldsymbol{c}_l$ as Eqn.7 and update $\boldsymbol{c}_l$ in prototype set $\boldsymbol{c}$.
    Compute and update prototypical loss $J_p(\phi_{\text{proto}})$ as Eqn.8.
  end
  [Inner Loop]
  Perform SGD on ϕ_{pred} to minimize Eqn.10 as $\phi'_{\text{pred}} = \phi_{\text{pred}} - \alpha \nabla_{\theta} \mathcal{L}_{inner}(S; \theta)$.
  [Prototypical Query Sample Editing]
  for class l in prototype set $\boldsymbol{c}$ **do**
    Select N_{select} nearest examples to $\boldsymbol{c}_l$ from $S \cup D_{\text{Memory}}$.
    Update examples of class l in D_{Memory}.
  end
  $Q \leftarrow D_{\text{Memory}}$ OR randomly sample a batch of data from D_{Memory}
  [Outer Loop]
  Perform Adam update on θ' to minimize Eqn.11 as $\theta \leftarrow \theta - \beta \nabla_{\theta'} \mathcal{L}_{outer}(Q; \theta')$, where $\theta' = \phi_{\text{proto}} \cup \phi'_{\text{pred}}$.
  if all training data are seen **then**
    Stop Iteration
  end
end

4.2 Query Sample Selection

The synthetic prototypes serve as a reference for query sample selection. In each episode, we choose query samples from training batches and memory set D_{memory}. In particular, we choose top N_{select} instances for each class with the highest similarity score to their corresponding prototype. The similarity score is calculated via Euclidean distance. The newly selected samples replace past ones and update memory set. The query samples of $\mathcal{T}_k$ is denoted as Q_k, in which we expect that $(\boldsymbol{x}, \boldsymbol{y}) \in Q_k \sim p(\mathcal{T}_k)$. We propose two read functions to obtain query samples: (a) read *all* from memory; (b) read *randomly* from memory, where the amount of retrieval instances equals to the mini-batch size, b.

5 Meta-learning with Prototypical Query Sample Editing

The meta-objective in outer loop governs the optimization process of task-specific learning in inner loop. We edit query samples by a prototypical network (Snell et al., 2017) to revise meta-objective to CL objective. We control the variation on parameter space by the edited meta-objective, hereby solving CL problems.

5.1 Edited Meta-objective for Continual Learning

We edit query examples via prototypes for outer loop algorithm. Then, we formulate a new meta-objective edited by prototypical information as:

$$\begin{aligned}
&\min_{\theta_K} \mathbb{E}_{\mathcal{T}}[\mathcal{L}_{\mathcal{T},Q}(U_{\mathcal{T}_K,S}(\theta_K))] \\
&= \min_{\theta_K} \mathbb{E}_{\mathcal{T}}[\mathcal{L}_{\mathcal{T},Q}(\tilde{\theta}_K)] \\
&= \min_{\theta_K} \sum_{k=1}^{K} \mathbb{E}_{\mathcal{T}_k}[\mathcal{L}_{\mathcal{T}_k,Q_k}(\tilde{\theta}_K)] \\
&= \min_{\theta_K} \sum_{k=1}^{K} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim p(\mathcal{T}_k)}[\mathcal{L}_{CE}(\boldsymbol{x}, \boldsymbol{y}; \tilde{\theta}_K)] \\
&= \min_{\theta_K} \sum_{k=1}^{K} \mathbb{E}_{\mathcal{T}_k}[\mathcal{L}_{\mathcal{T}_k}(\tilde{\theta}_K)]
\end{aligned} \tag{9}$$

Evidently, the edited meta-objective has the same expected loss as that of CL as shown in Eqn.2, but different optimization parameters.

Prevent catastrophic forgetting: The consistent expected loss indicates that the meta-learning framework can depict a CL scenario. Given a learned parameter set $\tilde{\theta}_K$ heavily biased towards the distribution of the latest task, $p(\mathcal{T}_K)$, the objective is to perform well on all tasks, $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_K\}$. Hence, we interpret CL problems into a form of meta-learning.

Preserve fast adaptation: Different optimization parameters suggest different learning strategy. As shown in Eqn.2, the optimization parameters of CL objective are finetuned parameters $\tilde{\theta}_K$. It neglects the next optimal steps, potentially causing more update steps for learning different domains. Conversely, the optimization parameters of edited meta-objective are initial parameters θ_K. It learns an optimal initialization state and yields a minimal loss with few update steps on all seen tasks. Hence, it preserves fast adaptation ability of meta-learning while solving catastrophic forgetting.

6 Model

The proposed model f_θ consists of a representation learning network $h_{\phi_{\text{proto}}}$ with learnable parameters ϕ_{proto} and a prediction network $g_{\phi_{\text{pred}}}$ with learnable parameters ϕ_{pred}. It is described as $f_\theta(x) = g_{\phi_{\text{pred}}}(h_{\phi_{\text{proto}}}(x))$. The prototypical network is a single-hidden-layer feed-forward neural network on top of an encoder. The prediction network is a single linear layer followed by a softmax.

Meta-training. We sample m batches of examples instantaneously from the data stream D_{train} as support set S for inner loop. The inner loop loss $\mathcal{L}_{\text{inner}}$ includes the current task loss $\mathcal{L}_{\mathcal{T}_K}$, i.e., cross entropy loss $\mathcal{L}_{CE}$, and prototypical network loss J_P,

$$\begin{aligned}
\mathcal{L}_{\text{inner}}(S;\theta) &= \mathcal{L}_{CE}(S;\theta) + J_P(\phi_{\text{proto}}) \\
&= \mathcal{L}_{CE}(S;\phi_{\text{proto}} \cup \phi_{\text{pred}}) + J_P(\phi_{\text{proto}})
\end{aligned} \tag{10}$$

The inner loop algorithm only finetunes PN model $g_{\phi_{\text{pred}}}$ via SGD with learning rate α, $\phi'_{\text{pred}} = \phi_{\text{pred}} - \alpha\nabla_\theta \mathcal{L}_{\text{inner}}(S;\theta)$. In outer loop algorithm, both parameters ϕ_{proto} and ϕ'_{pred} are meta-trained on query set Q. The outer loop loss is,

$$\begin{aligned}
\mathcal{L}_{\text{outer}}(Q;\theta') &= \sum_{k=1}^{K} \mathbb{E}_{\mathcal{T}_k}[\mathcal{L}_{\mathcal{T}_k}(\theta')] \\
&= \sum_{k=1}^{K} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim p(\mathcal{T}_k)}[\mathcal{L}_{CE}(\boldsymbol{x}, \boldsymbol{y}; \theta')] \\
&= \mathcal{L}_{CE}(Q;\theta')
\end{aligned} \tag{11}$$

where $\theta' = \phi_{\text{proto}} \cup \phi'_{\text{pred}}$. The optimizer of the outer loop algorithm is Adam optimizer with a learning rate β. The optimization process is $\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}_{\text{outer}}(Q; \theta')$, where the gradients are computed with respect to the initial parameters θ. To reduce computationally expensive, we use the first-order approximation, namely FOMAML (Finn et al., 2017). The gradients are computed with respect to the finetuned parameters θ'. Then, the outer loop optimization process is $\theta \leftarrow \theta - \beta \nabla_{\theta'} \mathcal{L}_{\text{outer}}(Q; \theta')$. The meta-training process is shown in Algorithm 1.

Meta-inference. We randomly sample m batches of examples from memory set D_{Memory} as the support set S for inner loop. We perform SGD on ϕ_{pred} to minimize cross entropy loss $\mathcal{L}_{CE}(S; \theta)$ as $\phi'_{\text{pred}} = \phi_{\text{pred}} - \alpha \nabla_\theta \mathcal{L}_{\text{inner}}(S; \theta)$. Then, we output the predication on the test samples $\boldsymbol{x}_{\text{test}}$ as $\hat{\boldsymbol{y}}_{\text{test}} = f_\theta(\boldsymbol{x}_{\text{test}}) = g_{\phi'_{\text{pred}}}(h_{\phi_{\text{proto}}}(\boldsymbol{x}_{\text{test}}))$.

7 Experiments

7.1 Datasets

We use the collection of text classification datasets from (Zhang et al., 2015) [1], including AGNews (news classification; 4 classes), Yelp (sentiment analysis; 5 classes), Amazon (sentiment analysis; 5 classes), DBpedia (Wikipedia article classification; 14 classes) and Yahoo (questions and answers categorization; 10 classes). Following prior work (de Masson d'Autume et al., 2019; Holla et al., 2020), we use the balanced version of the collection and merge the classes of Yelp and Amazon. Thus, we have 33 classes in total. We randomly sample 115,000 training examples and 7,600 test examples from each of the datasets. Each dataset is seen as a separate learning task.

Table 1: Input Datasets Orders

No.	Dataset Orders
1	Yelp → AGNews → Amazon
2	Yelp → Amazon → AGNews
3	Amazon → Yelp → AGNews
4	Amazon → AGNews → Yelp
5	AGNews → Yelp → Amazon
6	AGNews → Amazon → Yelp
7	Yelp → AGNews → DBpedia → Amazon → Yahoo
8	DBpedia → Yahoo → AGNews → Amazon → Yelp
9	Yelp → Yahoo → Amazon → DBpedia → AGNews
10	AGNews → Yelp → Amazon → Yahoo → DBpedia

[1] http://goo.gl/JyCnZq

7.2 Baselines

- **MAML-SEQ** is a sequential learning model using the same meta-learning (Finn et al., 2017) framework as ours without memory replay.
- **Replay** is a sequential learning model with episodic experience replay. It applies the common-used write and read mechanisms, i.e., write all and randomly read.
- **A-GEM** (Chaudhry et al., 2019) with episodic memory replay is a commonly-used lifelong language learning baseline, where catastrophic forgetting is prevented by gradient constraints.
- **OML-ER** (Holla et al., 2020) uses the same online meta-learning framework as ours but with episodic memory replay. Its default setting is writing all and randomly read.
- **PMR** (Ho et al., 2021) uses the same prototypical sample selection criteria as ours but use episodic memory replay.

7.3 Experimental Setup

We define our low resource scenario as insufficient training examples and strict memory constraints. We reduce the size of the training set to its **10%**, i.e., 11,500 per task. All models are trained in one epoch. We consider the effect of different orders of datasets. We conduct experiments on all six possible orders of three datasets, i.e., Yelp, AGNews and Amazon, as shown in Table 1. Following prior work (de Masson d'Autume et al., 2019; Holla et al., 2020), we also run experiments on all 5 tasks, using 4 possible orders.

The example encoder is a pre-trained ALBERT-Base-v2 (12M) from Hugging Face Transformers (Lan et al., 2020), where we truncate the input sequence length to 200. For meta-learning models, we use SGD as their inner loop optimizer with a learning rate, $\alpha = 1e^{-3}$, and Adam as their outer loop optimizer with a learning rate, $\beta = 3e^{-5}$. For A-GEM (Chaudhry et al., 2019) and Replay, we use Adam as their optimizer with learning rate, $3e^{-5}$. The number of mini-batches in each epoch, m = 5. Models with prototypical network samples data uniformly at random without replacement, where batch sizes depend on the number of classes from the training sets, e.g., b = 20 for AGNews and b

Table 2: Accuracy on 3 Tasks with 6 Different Orderings

Method	Memory Constraint (B)	Order Index (1)	(2)	(3)	(4)	(5)	(6)	Average Accuarcy
MAML-SEQ	No data saved	34.6	38.8	32.4	29.3	22.6	28.0	30.9 ± 5.6
AGEM	Unlimited, **100**% seen data	38.4	29.8	29.9	40.4	37.3	37.8	35.6 ± 4.6
Replay		42.5	29.8	29.6	46.5	40.4	38.7	37.9 ± 6.9
OML-ER		47.8	25.4	31.4	40.8	**48.5**	39.2	38.9 ± 9.1
PMR	5 per class, **0.13**% seen data	26.9	26.4	21.6	26.3	19.8	25.2	24.4 ± 3.0
Ours.(*all*)		**56.3**	**53.6**	**49.4**	**54.2**	47.9	**45.6**	$\mathbf{51.2 \pm 4.2}$
Ours.(*random*)		52.1	47.1	41.4	49.0	43.0	39.1	45.3 ± 4.9

Table 3: Per-task and Overall Forgetting on 3 Tasks

Order Index	Method	Yelp $A_{\text{CL},1}$	Yelp F_1	AGNews $A_{\text{CL},2}$	AGNews F_2	Amazon $A_{\text{CL},3}$	Amazon F_3	Overall Forgetting
Order (1)	OML-ER	46.8	3.26	49.2	36.8	47.3	3.2	43.3
	Ours.(*all*)	44.7	**-1.3**	79.5	**2.8**	44.5	**-4.9**	**-3.4**
	Ours.(*random*)	41.3	0.4	74.3	6.4	40.7	-4.5	2.3
Order Index	Method	AGNews $A_{\text{CL},1}$	AGNews F_1	Yelp $A_{\text{CL},2}$	Yelp F_2	Amazon $A_{\text{CL},3}$	Amazon F_3	Overall Forgetting
Order (5)	OML-ER	58.4	27.7	43.4	6.8	43.9	6.6	41.2
	Ours.(*all*)	68.0	14.3	38.2	**5.2**	37.6	**2.0**	**21.5**
	Ours.(*random*)	74.1	**6.6**	26.9	14.8	28.0	8.2	29.6

= 25 for Yelp and Amazon. We set $N_S = 3 * m$ and $N_Q = 2 * m$ for prototypical loss computation. Other models utilise the random sampler, which randomly samples examples with batch size, b = 25. The write rate p_{write} is set to 1. All models are executed on a Linux platform with 8 Nvidia Tesla A100 GPU and 40 GB of RAM. All experiments are performed using PyTorch (Paszke et al., 2019).

7.4 Evaluation Metrics

We evaluate the models after learning all tasks. The evaluation metrics are accuracy and forgetting. Let $A_{\text{CL},k}$ be the macro-averaged accuracy on $\mathcal{T}_k$, the overall accuracy after learning a sequence of K tasks is $\text{ACC} = \frac{1}{K}\sum_{k=1}^{K} A_{\text{CL},k}$. Let $A_{\text{single},k}$ be the accuracy of learning one task $\mathcal{T}_k$. The forgetting on a sequence of K tasks is $F = \sum_{k=1}^{K} F_k = \sum_{k=1}^{K} A_{\text{single},k} - A_{\text{CL},k}$, where F_k is the forgetting on task $\mathcal{T}_k$ and describes how the updated parameters affect the performance on $\mathcal{T}_k$ after sequential learning all tasks.

7.5 Results

We run experiment on all 6 possible orders of training sets while learning a sequence of 3 tasks. We record the average of 3 best results from 5 trials.

Accuracy. As shown in Table 2, our model with read *all* function, obtains the best results in most of the training sequences, except for Order (5), which is only 0.6% smaller than the best result. Given downsized training sets, it maintains average accuracy to more than 50%, suggesting its fast adaptation ability. Its standard deviations of the average accuracy is 4.2%, which is smaller than most baselines. It shows a good robustness in terms of training set orders. The proposed method with read *random* function has the second-highest average accuracy. Under the same memory constraint, PMR is inferior to our models by at least 20%. Remarkably, both of the proposed models even outperform models with unlimited memory budget. The results demonstrate the superiority of our model in a low resource scenario.

Forgetting Measurements. Evaluation on forgetting shows model's ability of knowledge retention, but not directly indicating model performance. A negative value of forgetting suggests accuracy improvements. As shown in Table 2, our models obtain the best performance on Order (1) and the worst performance on Order (5). We compare our models with all baselines on these two orders (see Table 8 in Appendix). We show the result of our models and a strong baseline, OML-ER, in Table 3. For Order (1), both of our models have the negative score of forgetting on Amazon. It demonstrates a positive forward knowledge transfer occurs. All models obtain a relatively high forgetting score on AGNews. It is reasonable considering its distribution is quite different from Yelp and Amazon. As for Order (5), the last two tasks, Yelp and Ama-

Table 4: Per-task and Overall Accuracy with Different Memory Constraints

Memory Constraint (B)	Method	Yelp	AGNews	Amazon	Overall Accuracy
$B = 27$	OML-ER	**49.8**	19.4	**49.1**	39.4
$B = 27$ (3 per class)	Ours.(all)	44.0	**67.6**	43.8	**51.8**
	Ours.(random)	36.2	63.6	35.1	44.9
$B = 45$	OML-ER	**48.0**	13.1	**49.3**	36.8
$B = 45$ (5 per class)	Ours.(all)	44.7	**79.5**	44.5	**56.3**
	Ours.(random)	41.3	74.3	40.7	52.1
$B = 63$	OML-ER	52.7	24.6	50.3	42.5
$B = 63$ (7 per class)	Ours.(all)	**55.1**	**86.2**	**52.2**	**64.6**
	Ours.(random)	40.9	69.3	39.3	49.8
$B = 34,500$ (Unlimited)	OML-ER	46.8	49.2	47.3	47.8
	Ours.(all)	–	–	–	–
	Ours.(random)	**55.1**	**86.2**	**52.2**	**64.6**

Table 5: Comparison of Various Query Sample Selection Methods

Query Sample Selection Method		Yelp	AGNews	Amazon	Overall Accuracy
Random		43.6	69.4	43.7	52.2
Prototypical Sample Selection	Diversity (Ours.)	**44.7**	**79.5**	**44.5**	**56.3**
	Uncertainty	28.2	47.8	27.8	34.6

zon, are both from product reviews. As a result, OML-ER has enough iterations to perform memory replay on AGNews. Its accuracy on AGNews improves by 9.1%. our methods use few samples to represent prior task, the increase of iterations make them converge too well to generalize. Thereby, our model with read *all* function suffers over-fitting problems, especially on AGNews. Read *random* function lessens this problem on AGNews, but degrades the performance on Yelp and Amazon. We consider it as the trade-off between representation and generalization. Still, both of our models outperform OML-ER and address catastrophic forgetting.

7.6 Further Analysis

We use Order (1) to perform further analysis. Note that Yelp and Amazon datasets are product reviews, while AGNews are news. The data distributions are different. Hence, we pay extra attentions to model performance on AGNews.

Memory Efficiency. We compare our models with all baselines (see Table 9 in the Appendix). We show the result of our models and a strong baseline, OML-ER, in Table 4. Our method with read *all* [2] function achieves highest overall performance given different memory constraints. Surprisingly, its results are even better than OML-ER without memory constraint by at most 16.8%. Our method with read *random* function is not as good as our read *all* method. It suggests that read *all* function is more beneficial when giving a strict memory limitation. The result indicates memory efficiency of the proposed method. Also, both of our models outperform OML-ER on AGNews by a large margin. It shows our models' ability of knowledge retention.

Effect of Query Sample Selection. Table 5 compares different query sample selection strategies. We consider two main strategies, i.e., random and prototypical sample selections. We further study two popular paradigms in active learning, i.e., diversity-based and uncertainty-based method. Our proposed method is considered as a diversity-based method. While, opting for examples far from prototypes is an uncertainty-based method. The read function is read *all*. In general, random selection is seen as an efficient selection criteria. Diversity-based method outperforms random selection by more than 4% and outperforms uncertainty-based method by more than 20%.

Effect of Query Sample Editing. We compare our approach to the mainstream CL method, i.e., episodic memory replay. The sample selection method is prototypical sample selection and write function writes all selected samples. Table 6 shows our approach surpasses episodic memory replay in two different read functions. For read *all* function, it is superior to episodic memory replay by nearly 20%. The performance on AGNews exceeds episodic memory replay by more than 44%. The reason behind this bad performance of episodic

[2] Ours.(*all*) with unlimited memory means writing all data to memory and replaying all data, which is not possible for large scale dataset and can result in out-of-memory errors.

Table 6: Comparison Between Episodic Memory Replay and Our Method

Continual Leaning Strategy	Read Function	Yelp	AGNews	Amazon	Overall Accuracy
Episodic Memory Replay	All	34.8	34.9	34.2	34.6
Ours.(*all*)		**44.7**	**79.5**	**44.5**	**56.3**
Episodic Memory Replay	Random	32.8	19.5	31.6	27.8
Ours.(*random*)		**41.3**	**74.3**	**40.7**	**52.1**

Table 7: Accuracy on 5 Tasks with 4 Different Orderings

Method	No. of Training Samples	Memory Constraint	Order Index (5)	(6)	(7)	(8)	Average Accuracy
A-GEM†	575,000	575,000 (Unlimited)	70.7	65.9	67.5	63.6	66.9 ± 3.0
MbPA++†			70.8	70.9	70.2	70.7	70.7 ± 0.3
REPLAY‡			69.5	66.2	65.2	68.3	67.3 ± 2.0
OML-ER‡			75.4	76.5	75.4	75.4	$\mathbf{75.7 \pm 0.6}$
PMR		165 (**0.03**% seen data)	61.2	65.7	66.1	55.9	$\mathbf{62.2 \pm 4.8}$
Ours.(*all*)			54.2	64.1	64.6	48.1	57.8 ± 8.0
Ours.(*random*)			59.2	61.6	64.6	51.3	59.2 ± 5.7
A-GEM	57,500	57,500 (Unlimited)	24.7	33.2	25.6	22.8	26.6 ± 4.7
Replay			31.0	50.8	38.8	37.2	39.5 ± 8.3
OML-ER			43.7	53.3	44.2	39.4	45.2 ± 5.8
PMR		165 (**0.03**% seen data)	2.6	45.2	5.0	5.7	14.6 ± 20.4
Ours.(*all*)			56.8	60.7	65.2	51.3	$\mathbf{58.5 \pm 5.9}$
Ours.(*random*)			50.6	59.5	48.7	36.1	48.7 ± 9.6

memory replay is its insufficient training iterations. Consequently, it is prone to forgetting. Similarly, for read *random* function, our method outperforms episodic memory replay by a large margin. In addition, high replay frequency often leads to over-fitting problem, especially when the amount of memory samples is small. Our approach revisits a small amount of past samples every iteration. But it still maintains a more than 52% overall accuracy.

Learning More Tasks and Examples. We run experiments on 5 datasets in 4 different orderings. For comparison, the experimental setup follows prior work (de Masson d'Autume et al., 2019)[3], where encoder is a pretrained BERT (Devlin et al., 2019). As shown in Table 7, given full training examples and strict memory constraint, both of our models are inferior to PMR by 3.0∼4.4%. The over-fitting problem occurs when the training iterations increases. Hence, our method with read *random* function performs better than that with read *all* function. When we downsize the training sets to their 10%, our models are superior, validating their fast adaptation abilities. Furthermore, PMR performs surprisingly well on Order (6), compared to its performance on other training set orders. The last two tasks in Order (6) are Yelp and Amazon. These two datasets are both product reviews and from the same domain. It increases the number of training iterations for PMR to learn different probability distributions via episodic memory replay. Thus, it results in a better accuracy.

[3] † Results obtained from (de Masson d'Autume et al., 2019) and ‡ Results obtained from (Holla et al., 2020).

8 Conclusion

In this paper, we propose an enhanced meta-learning framework for continual text classification model learning, where we use a prototypical network to edit query samples and adapt meta-objective for continual learning. The experimental results manifest that our method has an outstanding performance in a low resource scenario, i.e., insufficient training examples and strict memory constraints. It also validates our method ensures fast adaptation while preventing catastrophic forgetting.

9 Limitations

Our method works well in a low resource scenario, where the amount of training data is not plentiful and the memory budget is strictly limited. Its performance also relies heavily on pre-trained language models. In addition, the meta-learning framework we used, namely MAML, is the standard framework. The effect of different meta-learning frameworks should be studied. We leave this investigation to future work. Furthermore, we can extend our model to other NLP tasks.

References

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2019. Efficient lifelong learning with A-GEM. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13122–13131.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Stella Ho, Ming Liu, Lan Du, Longxiang Gao, and Yong Xiang. 2021. Prototypes-guided memory replay for continual learning. *CoRR*, abs/2108.12641.

Nithin Holla, Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. 2020. Meta-learning with sparse experience replay for lifelong language learning. *CoRR*, abs/2009.04891.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zhizhong Li and Derek Hoiem. 2016. Learning without forgetting. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pages 614–629. Springer.

M. McCloskey and N. J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999.

Abiola Obamuyide and Andreas Vlachos. 2019. Meta-learning improves lifelong relation extraction. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 224–229, Florence, Italy. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. 2019. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4077–4087.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. LAMAL: language modeling is all you need for lifelong language learning. *CoRR*, abs/1909.03329.

Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. 2019. Sentence embedding alignment for lifelong relation extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 796–806. Association for Computational Linguistics.

Zirui Wang, Sanket Vaibhav Mehta, Barnabás Póczos, and Jaime G. Carbonell. 2020. Efficient meta lifelong-learning with limited memory. In *Proceedings of the 2020 Conference on Empirical Methods in*

Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020, pages 535–548. Association for Computational Linguistics.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

Kang Zhao, Hua Xu, Jiangong Yang, and Kai Gao. 2022. Consistent representation learning for continual relation extraction. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3402–3411. Association for Computational Linguistics.

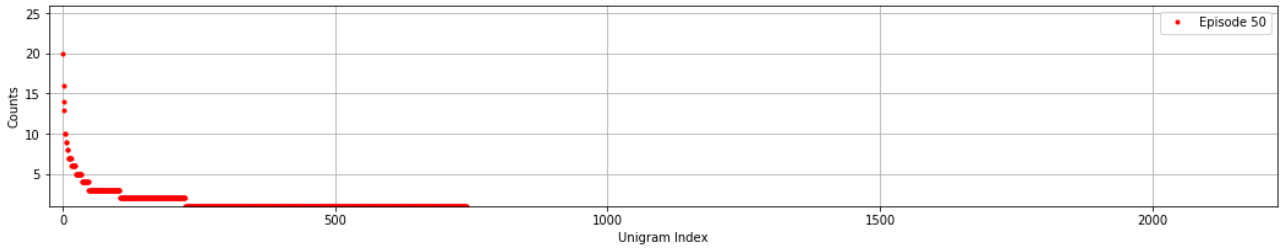
A Appendix

Table 8: Per-task and Overall Forgetting on 3 Tasks

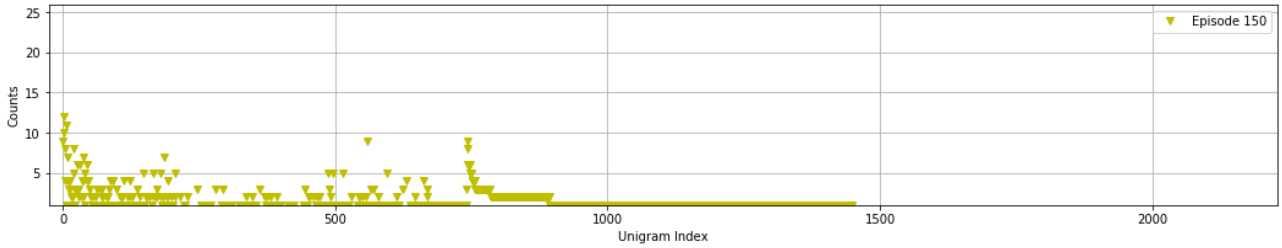
Order Index	Method	**Yelp**		**AGNews**		**Amazon**		Overall Forgetting
		$A_{\mathrm{CL},1}$	F_1	$A_{\mathrm{CL},2}$	F_2	$A_{\mathrm{CL},3}$	F_3	
Order (1)	MAML-SEQ	38.5	10.7	32.7	51.3	32.6	8.4	70.4
	AGEM	54.9	3.6	4.7	85.3	55.4	1.8	90.7
	Replay	56.7	1.8	13.3	76.7	57.5	-0.3	78.2
	OML-ER	46.8	3.26	49.2	36.8	47.3	3.2	43.3
	PMR	41.6	1.6	0.2	84.9	38.9	-0.9	85.6
	Ours.(*all*)	44.7	-1.3	79.5	2.8	44.5	-4.9	-3.4
	Ours.(*random*)	41.3	0.4	74.3	6.4	40.7	-4.5	2.3
Order Index	Method	**AGNews**		**Yelp**		**Amazon**		Overall Forgetting
		$A_{\mathrm{CL},1}$	F_1	$A_{\mathrm{CL},2}$	F_2	$A_{\mathrm{CL},3}$	F_3	
Order (5)	MAML-SEQ	27.9	56.1	20.4	28.8	19.4	21.6	106.5
	AGEM	0.1	89.9	55.8	2.7	56.1	1.1	93.7
	Replay	7.6	82.4	56.5	2.0	57.2	0.0	84.4
	OML-ER	58.4	27.7	43.4	6.8	43.9	6.6	41.2
	PMR	0.0	85.1	28.2	15.0	31.4	6.6	106.7
	Ours.(*all*)	68.0	14.3	38.2	5.2	37.6	2.0	21.5
	Ours.(*random*)	74.1	6.6	26.9	14.8	28.0	8.2	29.6

Table 9: Per-task and Overall Accuracy with Different Memory Constraints

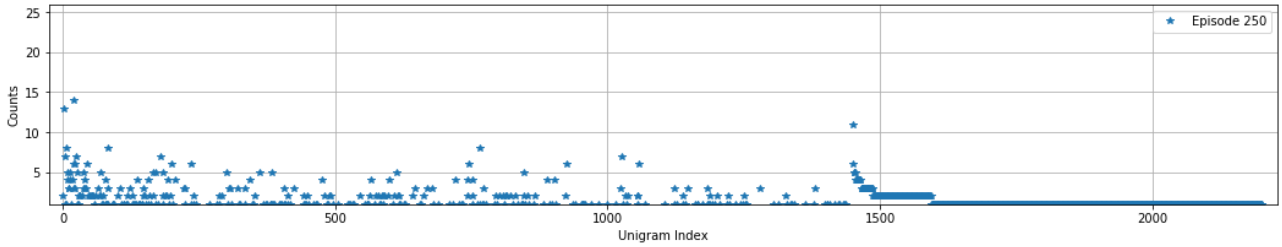
Memory Constraint (B)	Method	Yelp	AGNews	Amazon	Overall Accuracy
$B = 27$	AGEM	57.3	0.0	58.0	38.5
	Replay	57.1	1.6	57.3	38.7
	OML-ER	49.8	19.4	49.1	39.4
	PMR	39.1	0.4	39.6	26.4
$B = 27$ (3 per class)	Ours.(all)	44.0	67.6	43.8	51.8
	Ours.(random)	36.2	63.6	35.1	44.9
$B = 45$	AGEM	57.0	1.7	56.7	38.5
	Replay	57.3	2.1	57.8	39.0
	OML-ER	48.0	13.1	49.3	36.8
	PMR	41.6	0.2	38.9	26.9
$B = 45$ (5 per class)	Ours.(all)	44.7	79.5	44.5	56.3
	Ours.(random)	41.3	74.3	40.7	52.1
$B = 63$	AGEM	57.2	0.0	57.9	38.4
	Replay	56.9	3.3	57.5	39.3
	OML-ER	52.7	24.6	50.3	42.5
	PMR	38.4	3.6	36.1	26.0
$B = 63$ (7 per class)	Ours.(all)	55.1	86.2	52.2	64.6
	Ours.(random)	40.9	69.3	39.3	49.8
$B = 34,500$ (Unlimited)	AGEM	54.9	4.7	55.4	38.4
	Replay	56.7	13.3	57.5	42.5
	OML-ER	46.8	49.2	47.3	47.8
	PMR	–	–	–	–
	Ours.(all)	–	–	–	–
	Ours.(random)	55.1	86.2	52.2	64.6



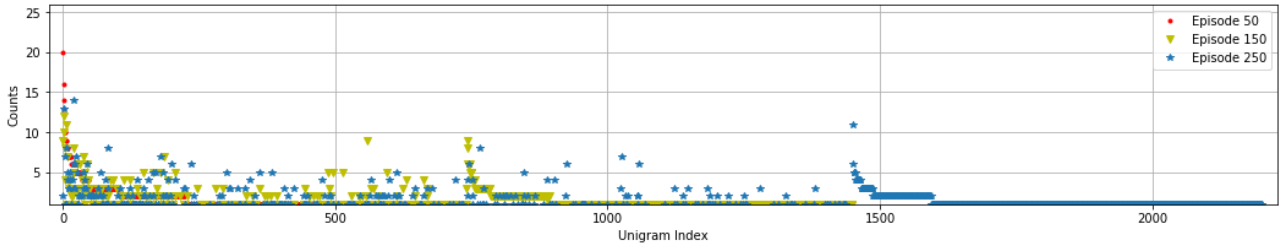
(a) Unigram distribution in episode 50



(b) Unigram distribution in episode 150



(c) Unigram distribution in episode 250



(d) Comparison of unigram distributions

Figure 1: The unigram distribution of the selected query samples. For simplicity, X-axis is the unigram index and Y-axis is the number of each unigram, where it is in the range $[\mathbf{1}, \mathbf{26})$. Note that the size of selected sample set is fixed.