Learn or Recall? Revisiting Incremental Learning with Pre-trained Language Models

Anonymous ACL submission

Abstract

Incremental Learning (IL) has been a long-standing problem in both vision and Natural Language Processing (NLP) communities. In recent years, as Pre-trained Language Models (PLMs) have achieved remarkable progress in various NLP downstream tasks, utilizing PLMs as backbones has become a common practice in recent research of IL in NLP. Most assume that catastrophic forgetting is the biggest obstacle to achieving superior IL performance and propose various techniques to overcome this issue. However, we find that this assumption is problematic. Specifically, we revisit more than 20 methods on four classification tasks (Text Classification, Intent Classification, Relation Extraction, and Named Entity Recognition) under the two most popular IL settings (Class-Incremental and Task-Incremental) and reveal that most of them severely underestimate the inherent anti-forgetting ability of PLMs. Based on the observation, we propose a frustratingly easy method called SEQ* for IL with PLMs. The results show that SEQ* has competitive or superior performance compared with state-of-the-art (SOTA) IL methods yet requires considerably less trainable parameters and training time. These findings urge us to revisit the IL with PLMs and encourage future studies to have a fundamental understanding of the catastrophic forgetting in PLMs. The data, code and scripts are in the supplementray material and will be publicly available [1].

1 Introduction

Learning knowledge incrementally without much forgetting is an essential ability of human beings but still an unsolved challenge for neural networks in achieving human-level intelligence (French, 1999). Incrementally learning a sequence of tasks can be formulated into the paradigm of Incremental Learning (IL) and has been impeded by catastrophic forgetting (Kirkpatrick et al., 2017). Catastrophic forgetting refers to neural networks forgetting previous knowledge after learning new tasks (McCloskey and Cohen, 1989).

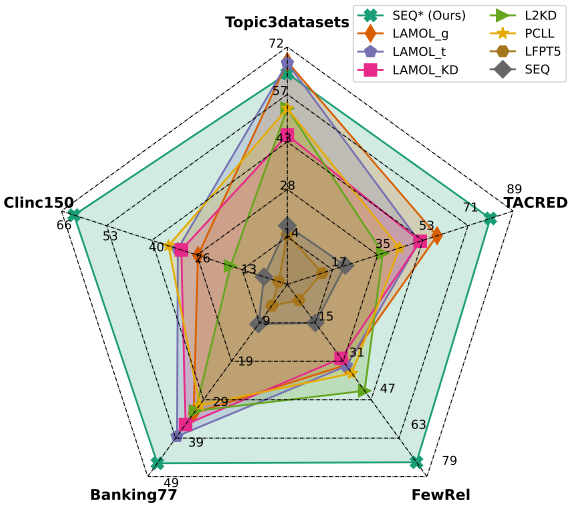


Figure 1: The comparison between the proposed SEQ* and SOTA IL methods on five class-incremental tasks. We report the average accuracy after learning the final task. The detailed results are provided in Table 1.

Recent years have witnessed significant breakthroughs in Pre-trained Language Models (PLMs) in vision and NLP tasks. Most recent studies of IL use PLMs as the backbone and design various methods for alleviating catastrophic forgetting in NLP tasks. However, is forgetting really catastrophical in PLMs? More specifically, how can we quantify forgetting and how much knowledge is forgotten in various IL scenarios when using various backbones and methods on various tasks?

To answer the above question, we carry out extensive experiments to explore forgetting in more than 20 methods on four classification tasks (Text Classification, Intent Classification, Relation Extraction, and Named Entity Recognition)

[1] Anonymous URL: https://anonymous.4open.science/r/code-for-learn-or-recall-B226

under the two most popular IL settings (Class-Incremental and Task-Incremental) with various model architecture (encoder only and decoder only) and scales (from 19M to 1.21B number of parameters). Through extensive experiments, we have several major findings:

- The popular assumption that PLMs suffer from catastrophic forgetting does not hold. Even under sequential fine-tuning (SEQ), the PLMs maintain the knowledge without much forgetting (Sec. 3.2). From the probing perspective, most existing IL methods do not learn incremental knowledge for PLMs (Sec. 4.2).

- By combining SEQ with simple strategies (Sec. 4.1), we propose SEQ* and find that SEQ* has competitive or even superior performance than SOTA IL methods (Figure 1, Sec. 4.2).

- The inherent anti-forgetting ability of PLMs comes from both the pre-training stage as well as the architecture of Transformer (Sec. 3.4). Randomly initialised PLMs learn incrementally when SEQ is performed on a sequence of tasks.

- The forgetting of SEQ is due to the deviation of the classifier from the PLM rather than the loss of old knowledge in the PLM. (Sec. 3.5).

Our study urges the NLP community to revisit and deepen the understanding of the forgetting in PLMs.

2 Experimental Settings

2.1 Problem formulation

Formally, the goal of IL is to learn a model $f_\theta : \mathbf{x} \rightarrow y \in \mathcal{Y}$ from a sequence of tasks $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_T\}$, where the t-th task $\mathcal{D}_t = \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}$ contains input samples $\mathbf{x}_i^t \in \mathcal{X}_t$ and labels $y_i^t \in \mathcal{Y}_t$. In Class-Incremental Learning (CIL), the label sets of different tasks are exclusive: $\mathcal{Y}_1 \cap \mathcal{Y}_2 \cdots \mathcal{Y}_T = \emptyset$, and the task id is unknown during inference. In Task-Incremental Learning (TIL), the label sets of different tasks may be overlapping: $\mathcal{Y}_1 \cap \mathcal{Y}_2 \cdots \mathcal{Y}_T \neq \emptyset$, and the task id is required during inference. In general, CIL is much more challenging than TIL because PLMs suffer from inter-task forgetting much more seriously than intra-task forgetting (Tao et al., 2023a). Appendix A provides detailed description and evaluation metrics.

2.2 Tasks and Datasets

We consider four types of downstream tasks in our experiments: Text classification, intent classification, relation extraction, and named entity recognition. We use the following eight datasets: Topic3Datasets (containing AGNews, DBPedia, and YaHoo (Zhang et al., 2015)) for text classification; Clinic150 (Larson et al., 2019) and Backing77 (Casanueva et al., 2020) for intent classification; FewRel (Han et al., 2018) and TACRED (Zhang et al., 2017) for relation extraction; OntoNotes5 (Hovy et al., 2006), I2B2 (Murphy et al., 2010), Few-NERD (Ding et al., 2021) for named entity recognition. Detailed descriptions are provided in Appendix B.

2.3 Backbones

We consider two popular architectures as backbone PLMs: encoder-only and decoder-only. For encoder-only backbones, we use bert-base-cased and bert-large-cased (Devlin et al., 2019), the most popular choices in previous IL studies. The encoder-only backbones are typically used as discriminant models, and linear layers are added for downstream tasks. We use the GPT2 (Radford et al., 2019) and Pythia suite (Biderman et al., 2023) for decoder-only backbones. Pythia is based on GPT-NeoX (Black et al., 2022), which contains 8 model sizes and 154 pre-training checkpoints, enabling research in interpretability and learning dynamics. The decoder-only backbones are typically used as generative models, and no additional linear layers are required since the output target is natural language. The detailed description is provided in Appendix C.

3 Revisiting the Forgetting from the Probing Perspective

3.1 How to Measure the Forgetting in PLMs?

This subsection describes how to measure the forgetting inside PLMs during IL. Specifically, we utilize the probing technique, an effective method to evaluate the representation ability of backbones on target tasks (Chen et al., 2023a; Tao et al., 2023b; Davari et al., 2022; Wu et al., 2021).

To probe the knowledge in PLMs for all tasks in IL, we add *probing classifiers* on top of the PLM and train the probing classifiers on *all tasks* in

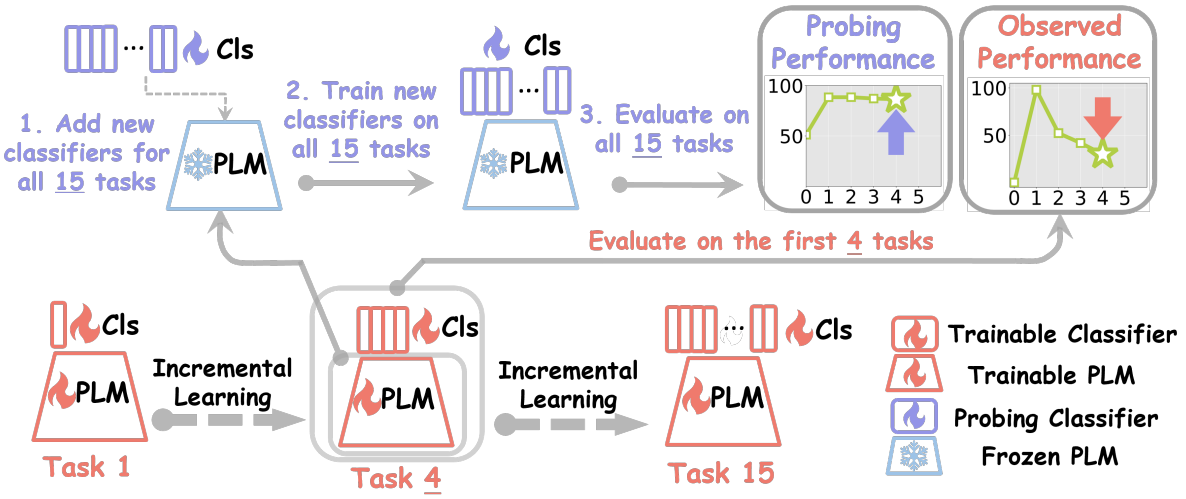


Figure 2: An illustration of how we obtain the probing and the observed performance of the model when learning the fourth task if there are a total of 15 tasks. The observed performance is used as a metric of forgetting in existing studies. The probing performance indicates how the encoder forgets. However, it is overlooked by previous studies.

IL. Then, we evaluate the PLM and the probing classifiers on all tasks and obtain the *probing performance*. The probing performance is the upper bound performance when the classifiers do not forget. For clarity, the performance evaluated with the original model is called the *observed performance*. We note that measuring probing performance will not affect the training process of IL since the backbone PLM is frozen when training probing classifiers. Furthermore, the original classifiers only predict the classes of *learned* tasks, while the probing classifiers predict the classes of *all* tasks in IL. We provide an illustration in Figure 2 and the formal definition in Appendix A.

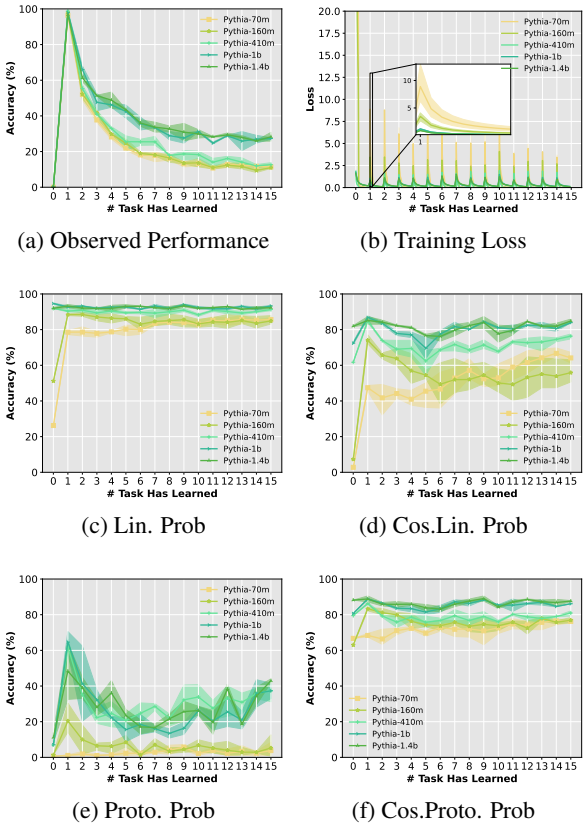
We consider four metrics for our probing study: linear probing, cosine linear probing, prototype probing, and cosine prototyping. Linear probing has been widely adopted in previous works. In linear probing, the probing classifier is a linear layer. Cosine linear probing adopts a cosine linear layer as the probing classifier. Specifically, the logits are computed as the cosine similarities between classifier weights and extracted features. Hou et al. (2019) show that utilizing cosine linear layers mitigates bias towards new classes in IL. In prototype probing, the probing classifier is a linear layer whose weight matrix is calculated as the class feature centres. Previous IL studies (Zhou et al., 2023; Chen et al., 2023b; Ma et al., 2023) show that using class feature centres as prototypes for classification is effective. Cosine prototype probing further utilizes cosine normalization when calculating logits. Further discussion is provided in Appendix D.1.

3.2 Is Sequential Fine-tuning Really the Lower Bound?

Sequential fine-tuning (SEQ) has long been regarded as the lower bound of IL. In this subsection, we revisit SEQ from the probing perspective, and we find that SEQ is severely underestimated when using PLMs for IL.

The backbone was small and randomly initialized in early studies exploring IL (Kirkpatrick et al., 2017; French, 1999; McCloskey and Cohen, 1989). They find that SEQ usually results in models forgetting all previous knowledge when learning new tasks. Recent IL studies in NLP (Razdaibiedina et al., 2023; Zheng et al., 2022; Huang et al., 2021; Sun et al., 2019) also observe that SEQ leads to worse performance. However, in the era of PLMs, fine-tuning has proven to be effective for adapting PLMs to different domains or downstream tasks (Aghajanyan et al., 2020; Devlin et al., 2019; Radford et al., 2018). If fine-tuning really causes PLMs to forget nearly all previous knowledge in IL, it should also cause PLMs to forget all pre-trained knowledge when adapting to new tasks. Obviously, this assumption is not true since fine-tuning is still effective for PLMs (OpenAI, 2023).

The observed and probing performance on class-incremental intent classification with generative models are summarized in Figure 3. The results on other IL settings, downstream tasks and backbones are in Appendix D.2. Figure 3a shows that the observed performance drops dramatically from approximately 98% to 10% as more new tasks are



(a) Observed Performance (b) Training Loss

(c) Lin. Prob (d) Cos.Lin. Prob

(e) Proto. Prob (f) Cos.Proto. Prob

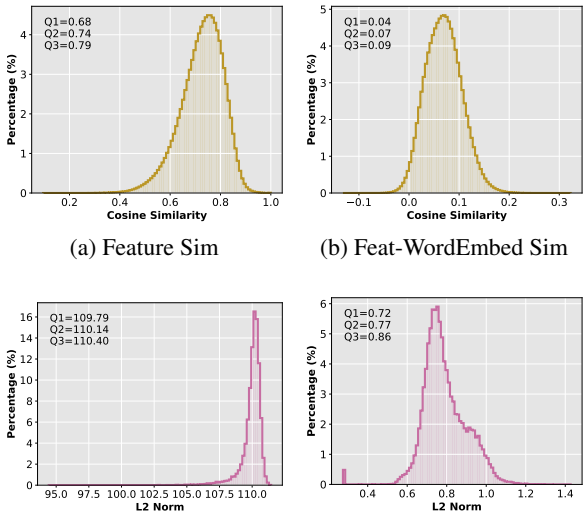
Figure 3: The observed and probing performance on Class-Incremental Intent Classification. The dataset is Clinic150. The backbones are generative models. (a)(b) are the observed performance and training loss during IL training. (c)-(f) are the probing performance when different metrics are adopted.

learned, in line with our understanding of catastrophic forgetting. However, Figure 3c describes an entirely different phenomenon. The PLMs achieve high probing performance after learning the first task. And the linear probing performance has barely decreased since the second task. In other words, PLMs preserve the knowledge to classify all 15 tasks even when adapting to only new tasks sequentially. This phenomenon is contradictory to what we know about catastrophic forgetting and SEQ.

Indeed, the probing performance is high since all tasks' data is available when training the probing classifiers, while the observed performance is poor since the original classifiers only train on the data from the current task. However, with the above observation, we can boost the observed performance with simple strategies, which will be described in Section 4.1.

3.3 What is the Best Metric for Probing Performance?

In Figure 3, we find that the ranking of four probing metrics is as follows: linear > cosine linear, cosine prototype > prototype probing. This subsection will explain why linear probing is the best metric for probing study.



(a) Feature Sim (b) Feat-WordEmbed Sim

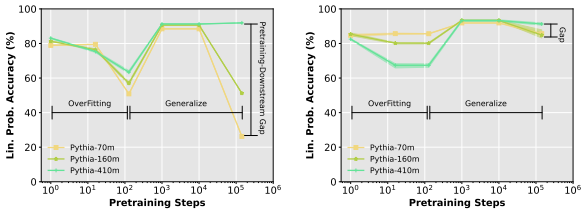
(c) Feature Norm (d) WordEmbed Norm

Figure 4: The histogram of features and different embeddings of Pythia-410m. The features are calculated on the training set of Clinic150, and the output word embeddings are loaded from pre-trained weights. The class embeddings refer to the row vectors of the weight matrix in the probing classifier on Clinic150. The class prototypes refer to the class feature centres estimated on the training set of Clinic150.

First, we need to understand what the features (i.e., last hidden states), word embeddings of PLMs and the class embeddings in probing classifiers "look like". The detailed description is in Appendix D.3. The histograms of the L2 norm and the cosine similarity of features, word embeddings and class embeddings are in Figure 4. Figure 4a shows that the features occupy a narrow cone in the vector space rather than being uniform in all directions, which has been discussed in (Ethayarajh, 2019). More surprisingly, Figure 4b shows that the learned (output) word embeddings are nearly orthogonal to the features. We infer that the cross-entropy loss encourages all word embeddings except the ground truth one to become farther away from the feature during pre-training. In other words, the cross-entropy loss encourages a large difference in logits, and the word embeddings to be orthogonal to the features in order to distinguish logits better.

Therefore, it is not surprising that linear probing has the best performance, considering that the word embedding layer is essentially a linear layer. From this point of view, it is also not surprising that the performance of prototype probing is poor since the prototypes (class feature centres) also fall in the narrow cone space, and it is not an optimal solution for distinguishing logits.

Then, why does cosine normalization degrade the performance of linear probing but improve prototype probing? Figure 4c and 4d are the L2 norm of the features and word embedings. We find that the norm of word embeddings has a larger discrepancy than features. It indicates that the norm of word embeddings contains the prior knowledge obtained from pre-training. Therefore, the cosine linear probing ignores the difference in the norm of features and thus has poorer performance compared with linear probing. For prototype probing, the prototype falls in a narrow cone space, and the similarity between the prototype and features is large and close to each other. In this case, cosine normalization can eliminate the interference of the norm and establish the relationship between logits and cosine similarity between features. We provide the detailed analysis and full results with different backbones in Appendix D.3. An illustration of different types of probing metrics is in Figure 7.



(a) Intent+Before SEQ (b) Intent+After SEQ

(c) RE+Before SEQ (d) RE+After SEQ

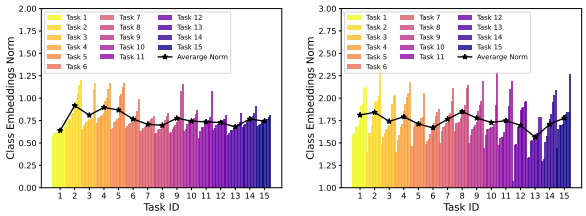
Figure 5: The linear probing performance on checkpoints with different pre-training steps. (a) and (b) are evaluated before and after incremental learning using SEQ. "Intent" and "RE" represent the model is evaluated on the Class-Incremental Intent Classification or Relation Extraction.

3.4 What is the Role of Pre-training in IL?

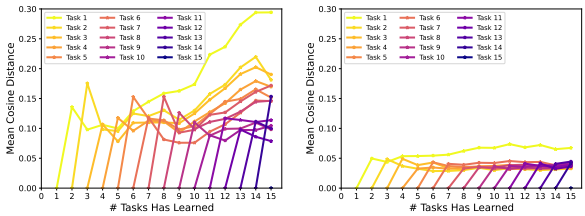
In this subsection, we reveal that the key to the anti-forgetting ability of PLMs lies in both the Transformers' architecture and the pre-training knowledge.

We evaluate the linear probing performance on checkpoints with a different number of pre-training steps: {0,16,128,1k,10k,143k(final)}. We load the pre-trained checkpoints (or randomly-initialized checkpoints at step 0) and evaluate their linear probing performance before and after IL using SEQ. Figure 5 shows two main phases in pre-training: overfitting and generalization. In the first phase (step 0 - step 128), the model memorizes the pre-training corpus, and the linear probing performance decreases. In the second phase (step 1k - step 143k), the model gradually learns the pre-training knowledge and the linear probing performance increases. However, when the model further generalizes to the pre-training corpus (step 10k - step 143k), the linear probing performance of small backbones (Pythia-70 m and 160m) decreases again due to the gap between pre-training and downstream tasks. This gap can be eliminated when adapting to downstream tasks (Figure 5a and Figure 3c). For larger backbones (Pythia-410 m, 1b, and 1.4b), the model can be adapted to new tasks directly without this gap.

Besides, we have the following interesting findings: (1) Pre-training indeed improves the linear probing performance in IL (Figure 5b and 5d). (2) Apart from pre-training, the architecture of the Transformer is also a key factor in the high linear probing accuracy during SEQ. When the downstream task is relatively simple, such as intent classification, even the randomly-initialized models achieve high linear probing performance (Figure 5b). Pre-training brings considerable improvements when the downstream task is more complex, such as relation extraction (Figure 5d). (3) More surprisingly, SEQ improves the linear probing performance of models from nearly all pre-training steps (Figure 5a v.s. 5a; Figure 5c v.s. 5d). This shows that Transformers' architecture can incrementally absorb new knowledge even when just sequential fine-tuning on new tasks. The detailed settings, visualization of features, and additional results on text classification are provided in Appendix E.



(a) Observed Classifier (b) Probing Classifier



(c) Observed Classifier (d) Probing Classifier

Figure 6: Comparison between the observed linear classifier and the linear probing classifier during SEQ on class-incremental intent classification. The backbone is Pythia-410m. (a)(b) show the average norm of the class embeddings of each task; (c)(d) show the average moving distance of the class embeddings of each task.

3.5 What is really forgotten in SEQ?

As discussed in Sec. 3.2, SEQ's linear probing performance showed little degradation or even improvement across all settings. Therefore, the reason for forgetting must lie in the classifier. In this subsection, we will take a closer look at the forgetting in classifiers.

Existing studies (Wu et al., 2019; Hou et al., 2019) find that the model tends to predict new classes during IL. They refer this phenomenon as the class imbalance problem between old and new classes. In SEQ, the class imbalance problem is more severe since only new classes are learned. We also observe that the logits of new classes are much larger than those of old classes in a SEQ model. Because both features and class embeddings determine the magnitude of logits, the features occupy a narrow cone space, and their norms are relatively close, we can infer that the forgetting is caused by either (1) the norm of class embeddings or (2) the cosine similarity between features and class embeddings.

For the first reason (i.e., class norm), we compare the class embedding norm between learned linear classifiers and linear probing classifiers in Figure 6a and 6b. Surprisingly, the class embedding norm of new tasks is not larger than those of old tasks in the observed classifier of SEQ. It indicates that the class norm is not the primary reason for the forgetting in SEQ.

For the second reason (i.e., cosine similarity), we compare the moving distance of class embeddings between the observed and probing classifiers in Figure 6c and 6d. The moving distance of the class embeddings of task t at task $t+k$ is computed as follows: (1) When the model finishes training on task t, we compute the cosine distance between all pairs of class embeddings from task t and class feature centres from *all tasks* and obtain a cosine similarity matrix $\mathbf{C}_t^t$. (2) When model finishes training on task $t+k$, we compute the cosine distance between all pairs of class embeddings from task t and class feature centres from *all tasks* and obtain a cosine similarity matrix $\mathbf{C}_{t+k}^t$. (3) Then, the moving distance of task t's class embeddings is calculated as the average absolute difference between the cosine similarity matrix $\mathbf{C}_t^t$ and $\mathbf{C}_{t+k}^t$. The moving distance measures how the class embeddings move relatively to *all* class feature centres since they have been learned. If the classifier does not forget a class, the distance from its class embeddings to all class feature centres should remain constant. In other words, its moving distance will be zero if the classifier does not forget how to classify this class with the features extracted by PLMs. We provide an illustration in Figure 25. The detailed settings, definition, and additional results with frozen bert-large-cased are provided in Appendix F.

Figure 6c and 6d show that the class embeddings of observed classifiers change significantly compared with those of probing classifiers. It indicates that the forgetting happens because the old class embeddings are pushed away from their initial and optimal position. The cosine similarity matrices are visualized in Figure 26.

4 Revisiting the SOTA Methods in IL

4.1 SEQ*: Boosting the Performance of SEQ

In this subsection, we propose SEQ* based on the findings about the forgetting in SEQ.

In the previous section, we have the following findings about SEQ: (F1) The PLMs do not learn new knowledge in SEQ about the downstream IL tasks; (F2) The PLMs achieve the highest probing performance once being adapted to downstream tasks, and there is little performance degradation when learning on more new tasks (See Figure 3c and Figure 5a); (F3) The classifier forgets dramati-

cally, while the PLMs do not. The reason is that the class embeddings are pushed away from the initial learned optimal position.

Therefore, we propose the following strategies for closing the gap between the probing and observed performance in SEQ: (S1) Freeze the PLMs after warm-up; (S2) Freeze the old classifiers when learning new tasks; (S3) Use cosine linear classifiers only when no old data is available in a CIL scenario. Otherwise, use linear classifiers; (S4, optional) Pre-allocate future classifiers. We call the method with the above strategies as SEQ*, and an illustration is provided in Figure 27.

The rationale for the above strategies is as follows: (S1) is proposed according to (F1). Furthermore, we propose to warm up (i.e., full-parameter fine-tuning) PLMs in only the first task according to (F2). In practice, warm-up onlt for 1-3 epochs brings considerable improvement across backbones and datasets. (S1) and (S2) preserve the relative position of class embeddings with respective to class feature centres to avoid the issue in (F3). When both PLMs and classifiers of old tasks are frozen, only the norm of new class embeddings may lead to the biased prediction towards new classes. Because cosine linear layers are not optimal for exploiting PLMs' knowledge, we propose (S3) to avoid bias prediction. In other words, we use linear classifiers for the TIL scenario and the CIL scenario where old data is stored. Finally, we propose (S4) for better forward compatibility (Zhou et al., 2022). (S4) is marked as an optional strategy since it requires additional information on the number of total tasks. Therefore, we report the two variants of SEQ*, i.e., w/ and w/o (S4), when comparing with SOTA methods. We provide detailed discussion and explanation in Appendix G.

4.2 Comparing SOTA methods with SEQ*

In this subsection, we evaluate SEQ* under extensive settings. Despite its simplicity, SEQ* has competitive or even superior performance in most settings.

We provide the result under the CIL scenario in Figure 1, Table 1 and 2 in the main manuscript due to the space limitation. We provide the introduction and training details of SOTA methods in Appendix H. The full results on other backbones, datasets, and IL settings are summarized in Table 7, 8, 10, 11, 12, 13, 9, 15 in Appendix I. All baselines and SEQ* use the same backbone PLM for IL. In all settings except for sentence-level classification tasks with discriminant backbones, SEQ* and all baselines store no old samples.

Table 1: Comparison between SOTA methods and SEQ* on sentence-level classification tasks. The backbone is Pythia-410m. The IL scenario is CIL. No old samples are stored for all models. *Lin*: use linear classifiers; *Cos*: use cosine linear classifiers; *FixB*: fix backbone PLMs; *FixC*: fix old classifiers; *FixBC*: fix both backbone PLMs and old classifiers; *W*: warm up backbone PLMs; *P*: pre-allocate future classifiers. The best and second best results are **bold** and underlined. The full result is in Table 7.

	Topic3Datasets	Clinic150	Banking77	FewRel	TACRED
	$\mathcal{A}_T$	$\mathcal{A}_T$	$\mathcal{A}_T$	$\mathcal{A}_T$	$\mathcal{A}_T$
LFPT5	16.78	3.48	7.98	5.52	7.60
L2KD	58.89	22.48	47.47	37.08	20.86
LAMOL_KD	49.94	41.99	52.60	25.77	29.03
LAMOL_g	**74.45**	35.43	48.40	28.10	32.70
LAMOL_t	74.05	43.37	57.00	28.44	28.81
PCLL	58.83	47.09	45.33	31.00	24.50
SEQ (Lin)	19.66	9.26	14.88	13.43	12.64
SEQ (Cos)	16.89	5.97	11.10	11.40	10.08
SEQ (FixB+Cos)	17.13	6.08	10.32	7.45	9.30
SEQ (FixC+Cos)	50.96	64.28	44.93	33.48	28.90
SEQ (FixBC+Cos)	53.18	62.72	44.09	33.58	28.02
SEQ (W+FixBC+Lin)	33.41	19.06	17.79	13.68	13.65
SEQ (P+W+FixBC+Lin)	33.70	27.20	15.09	17.08	14.54
SEQ* (W+FixBC+Cos)	50.77	75.96	53.76	46.12	36.55
SEQ* (P+W+FixBC+Cos)	70.56	**84.51**	**67.12**	**61.99**	**44.34**

Table 2: Comparison between SOTA methods and SEQ* on word-level classification tasks. The backbone is bert-base-cased. The IL scenario is CIL. No old samples are stored for all models. Other notation is the same as Table 1. The full result is in Table 13.

	Few-NERD	OntoNotes5	I2B2
	$\mathcal{A}_T$	$\mathcal{A}_T$	$\mathcal{A}_T$
SpanKL	18.26	40.10	6.12
OCILNER	18.44	39.99	27.27
ExtendNER	20.02	48.08	20.02
DLD	20.75	47.23	30.50
SelfTrain	23.46	51.08	23.60
RDP	27.08	50.45	40.38
CPFD	**34.65**	55.58	43.52
ICE_O	28.98	51.81	49.12
ICE_PLO	19.94	46.52	47.76
CFNER	27.70	58.07	35.42
SEQ (Lin)	2.97	4.38	5.26
SEQ (W+FixBC+Cos)	7.26	29.12	45.95
SEQ (P+W+FixBC+Cos)	3.17	29.70	47.10
SEQ* (W+FixBC+Lin)	28.13	66.99	71.76
SEQ* (P+W+FixBC+Lin)	28.21	**67.39**	**72.51**

From the results in Table 1 and 2, we have the following findings: (1) SEQ* shows comparable or better performance on all datasets. Using proper classifiers, fixing old classifiers, and pre-allocating future classifiers improve SEQ significantly. We highlight that we do not aim to show SEQ* achieves SOTA performance across all settings. Instead, we aim to demonstrate that SEQ* serves as a comparable baseline in most IL settings

and should be considered in further IL studies.

(2) SEQ* does not perform best when the PLM is required to absorb new knowledge, or there are overlaps between new and old tasks. For example, Few-NERD contains fine-grained entities, such as "Airport" and "Hotel", which PLMs may not have seen during pre-training. In Topic3Datasets, "Sci/Tech" and "Computers & Internet" belong to two different tasks. Intuitively, the PLM need to adjust the class boundary to avoid overlapping between classes.

Furthermore, we compare the linear probing performance between SEQ* and SOTA methods in Figure 3 and 14. The results show that the difference in the linear probing performance is small compared with the observed performance. The improvement between "BeforeIL" and "AfterIL" mainly comes from the adaptation from pre-training to downstream tasks (Figure 3,9,11,12). It explains why freezing PLMs after warm-up is effective. It also explains why prompt-based methods (Razdaibiedina et al., 2023; Wang et al., 2022) are effective even if only a tiny portion of the parameters are learned. Furthermore, the performance gap between SEQ* and linear probing performance still exists. The reason is that backward knowledge transfer from new tasks to old tasks is prohibited in SEQ*.

We compare the training time and the number of trainable parameters between SEQ* and SOTA methods in Table 4. SEQ* requires much less training time and trainable parameters for each task.

Table 3: The linear probing performance with Pythia-410m. Other settings are the same as Table 1.

	Clinic150		FewRel	
	Before IL	**After IL**	**Before IL**	**After IL**
SEQ (Lin)	91.05±0.65	91.08±0.20	52.18±0.50	77.39±0.28
L2KD		90.88±0.78		76.57±0.46
LAMOL_t		91.30±1.14		81.54±0.66
LAMOL_g		91.42±0.25		81.09±0.71
SEQ* (P+W+FixBC+Cos)		91.12±0.52		77.47±0.84

Table 4: The comparison of training time and trainable parameters for each task on Clinic150. †: The model after warm-up.

	Time (Min)	**# Trainable Params each Task**
PCLL	199	410M
L2KD	179	405M
LAMOL_KD	119	405M
LAMOL_t	70	405M
LAMOL_g	68	405M
SEQ*	24	10.24K†

5 Related Work

Previous studies have assessed catastrophic forgetting by measuring performance degradation on old tasks. However, there is limited understanding of probing performance in incremental learning. Davari et al. (2022) use linear probing to reveal that representations still experience significant drift due to parameter updates. Wu et al. (2021) conduct layer-wise probing studies on BERT, revealing catastrophic forgetting in the top and middle layers. They observed that, although BERT maintains high representational ability at the last incremental step, the classifier loses the ability to classify previously learned classes. Chen et al. (2023a) conduct linear probing on k-shot samples from the next task, revealing a strong correlation between retaining past information and learning efficiency on new tasks. Tao et al. (2023b) utilize linear probing to illustrate that BERT is inherently resilient to catastrophic forgetting, even without buffer data in task-incremental learning. In this study, we further investigate the influence of probing metrics, backbone and classifier architecture, pre-training steps, datasets, and IL methods on the probing performance. Furthermore, we analyze the forgetting of classifiers from the perspective of norm and cosine similarity. Finally, we propose simple but effective strategies for SEQ and conduct extensive experiments to validate its effectiveness.

6 Conclusion

Incremental learning is a key pillar of human cognition and intelligence. As PLMs have become popular in recent years, more and more IL studies adopt PLMs as the backbone model. However, we reveal that existing studies ignore the inherent anti-forgetting of PLMs and design methods based on a problematic assumption. Our findings encourage the IL community to revisit the assumption of catastrophic forgetting in PLMs and re-evaluate the proposed IL algorithms by comparing them with frozen-based methods such as SEQ*.

We suggest two future directions for IL with PLMs: (1) design IL benchmark where domain-specific knowledge is required; (2) design IL algorithms that update the knowledge of PLMs with limited time, computation cost and memory budget.

Limitations

There are two limitations of this study: (1) We only focused on the IL of classification tasks and did not

explore the forgetting of general forms of knowledge in PLMs; (2) We did not fully understand the internal mechanism of how PLMs incrementally learn the knowledge under SEQ.

Ethical Considerations

The ethical considerations of our research are carefully addressed to ensure compliance with relevant standards and transparency. To this end, we provide the following clarifications for reproducibility:

- We provide a detailed setting of our experiments.
- The source code, data, and scripts will all be publicly available.
- Our findings are in alignment with observed empirical outcomes.

References

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2020. Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations*.

Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. 2022. Learning fast, learning slow: A general continual learning method based on complementary learning system. In *International Conference on Learning Representations*.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin. Association for Computational Linguistics.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. 2020. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, Online. Association for Computational Linguistics.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–547.

Jiefeng Chen, Timothy Nguyen, Dilan Gorur, and Arslan Chaudhry. 2023a. Is forgetting less a good inductive bias for forward transfer? In *The Eleventh International Conference on Learning Representations*.

Xiudi Chen, Hui Wu, and Xiaodong Shi. 2023b. Consistent prototype learning for few-shot continual relation extraction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7409–7422, Toronto, Canada. Association for Computational Linguistics.

Yung-Sung Chuang, Shang-Yu Su, and Yun-Nung Chen. 2020. Lifelong language knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2914–2924, Online. Association for Computational Linguistics.

Li Cui, Deqing Yang, Jiaxin Yu, Chengwei Hu, Jiayang Cheng, Jingjie Yi, and Yanghua Xiao. 2021. Refining sample embeddings with relation prototypes to enhance continual relation extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 232–243, Online. Association for Computational Linguistics.

MohammadReza Davari, Nader Asadi, Sudhir Mudur, Rahaf Aljundi, and Eugene Belilovsky. 2022. Probing representation forgetting in supervised and unsupervised continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16712–16721.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2019. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2(6):2.

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, Volume 1 (Long and Short Papers)*, pages

4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021. Few-NERD: A few-shot named entity recognition dataset. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3198–3213, Online. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. 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 55–65, Hong Kong, China. Association for Computational Linguistics.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Xu Han, Yi Dai, Tianyu Gao, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2020. Continual relation learning via episodic memory activation and reconsolidation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6429–6440, Online. Association for Computational Linguistics.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4803–4809, Brussels, Belgium. Association for Computational Linguistics.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2019. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 831–839.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA. Association for Computational Linguistics.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Yufan Huang, Yanzhe Zhang, Jiaao Chen, Xuezhi Wang, and Diyi Yang. 2021. Continual learning for text classification with information disentanglement based regularization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2736–2746, Online. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv e-prints*, pages arXiv–1412.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. 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 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Minqian Liu and Lifu Huang. 2023. Teamwork is not always good: An empirical study of classifier drift in class-incremental information extraction. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2241–2257, Toronto, Canada. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Ruotian Ma, Xuanting Chen, Zhang Lin, Xin Zhou, Junzhe Wang, Tao Gui, Qi Zhang, Xiang Gao, and Yun Wen Chen. 2023. Learning "O" helps for learning more: Handling the unlabeled entity problem for class-incremental NER. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5959–5979, Toronto, Canada. Association for Computational Linguistics.

Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, Pascale Fung, and Zhiguang Wang. 2021. Continual learning in task-oriented dialogue systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages

7452–7467, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Natawut Monaikul, Giuseppe Castellucci, Simone Filice, and Oleg Rokhlenko. 2021. Continual learning for named entity recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13570–13577.

Shawn N Murphy, Griffin Weber, Michael Mendis, Vivian Gainer, Henry C Chueh, Susanne Churchill, and Isaac Kohane. 2010. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). *Journal of the American Medical Informatics Association*, 17(2):124–130.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Guangyue Peng, Tao Ge, Si-Qing Chen, Furu Wei, and Houfeng Wang. 2023. Semiparametric language models are scalable continual learners. *arXiv preprint arXiv:2303.01421*.

Chengwei Qin and Shafiq Joty. 2021. Lfpt5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. In *International Conference on Learning Representations*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6107–6122, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yijia Shao, Yiduo Guo, Dongyan Zhao, and Bing Liu. 2023. Class-incremental learning based on label generation. *arXiv preprint arXiv:2306.12619*.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. Lamol: Language modeling for lifelong language learning. In *International Conference on Learning Representations*.

Mingxu Tao, Yansong Feng, and Dongyan Zhao. 2023a. Can bert refrain from forgetting on sequential tasks? a probing study. In *The Eleventh International Conference on Learning Representations*.

Mingxu Tao, Yansong Feng, and Dongyan Zhao. 2023b. Can bert refrain from forgetting on sequential tasks? a probing study. In *The Eleventh International Conference on Learning Representations*.

Davoud Ataee Tarzanagh, Yingcong Li, Christos Thrampoulidis, and Samet Oymak. 2023. Transformers as support vector machines. *arXiv preprint arXiv:2308.16898*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Tongtong Wu, Massimo Caccia, Zhuang Li, Yuan-Fang Li, Guilin Qi, and Gholamreza Haffari. 2021. Pretrained language model in continual learning: A comparative study. In *International Conference on Learning Representations*.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 374–382.

Duzhen Zhang, Wei Cong, Jiahua Dong, Yahan Yu, Xiuyi Chen, Yonggang Zhang, and Zhen Fang. 2023a. Continual named entity recognition without catastrophic forgetting. *arXiv preprint arXiv:2310.14541*.

Duzhen Zhang, Hongliu Li, Wei Cong, Rongtao Xu, Jiahua Dong, and Xiuyi Chen. 2023b. Task relation distillation and prototypical pseudo label for incremental named entity recognition. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 3319–3329.

Duzhen Zhang, Yahan Yu, Feilong Chen, and Xiuyi Chen. 2023c. Decomposing logits distillation for incremental named entity recognition. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1919–1923.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark. Association for Computational Linguistics.

Yunan Zhang and Qingcai Chen. 2023. A neural span-based continual named entity recognition model. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Yingxiu Zhao, Yinhe Zheng, Zhiliang Tian, Chang Gao, Jian Sun, and Nevin L. Zhang. 2022. Prompt conditioned VAE: Enhancing generative replay for lifelong learning in task-oriented dialogue. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11153–11169, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Junhao Zheng, Zhanxian Liang, Haibin Chen, and Qianli Ma. 2022. Distilling causal effect from miscellaneous other-class for continual named entity recognition. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3602–3615, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, Liang Ma, Shiliang Pu, and De-Chuan Zhan. 2022. Forward compatible few-shot class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9046–9056.

Da-Wei Zhou, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. 2023. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *arXiv preprint arXiv:2303.07338*.

Appendix

A Problem Formulation

A.1 Overview of IL

Incremental Learning (IL) aims to learn a model on new tasks incrementally without forgetting previous knowledge. In this paper, we only consider classification tasks, which are popular and challenging settings in existing studies. Formally, IL aims to learn a model $f_\theta : \mathbf{x} \rightarrow y \in \mathcal{Y}$ from the sequence of tasks $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_T\}$, where the t-th task $\mathcal{D}_t = \{(\mathbf{x}_i^t, y_i^t)\}_{i=1}$ contains input samples $\mathbf{x}_i^t \in \mathcal{X}_t$ and labels $y_i^t \in \mathcal{Y}_t$. There are three popular scenarios in IL: Class-Incremental Learning (CIL), Task-Incremental Learning (TIL), and Domain-Incremental Learning (DIL). In CIL, the label sets of different tasks are exclusive: $\mathcal{Y}_1 \cap \mathcal{Y}_2 \cdots \mathcal{Y}_T = \emptyset$, and the task identity is unknown during inference. In TIL, the label sets of different tasks may be overlapping: $\mathcal{Y}_1 \cap \mathcal{Y}_2 \cdots \mathcal{Y}_T \neq \emptyset$, and the task identity is required during inference. In DIL, the label sets of different tasks are the same: $\mathcal{Y}_1 = \mathcal{Y}_2 = \mathcal{Y}_T$. Under the data replay setting, a buffer $\mathcal{M}$ is introduced for storing old representative instances. In the main experiments of this research, we consider the most challenging scenario, CIL, where catastrophic forgetting occurs most severely. The result on TIL is also reported when compared with state-of-the-art methods.

A.2 Evaluation Metric for IL

We adopt average accuracy (Chaudhry et al., 2018) as the metric for evaluation. Specifically, the average accuracy at task t is defined as the following

$$\mathcal{A}_t = \frac{1}{t}\sum_{i=1}^{t} a_{t,i}, \tag{1}$$

where $a_{t,i}$ represents the accuracy evaluated on the test set of task i after training the model incrementally from tasks 1 to t. The average accuracy indicates the performance on all learnt tasks. In the main manuscript, we report the average accuracy after learning the final task, i.e., $\mathcal{A}_T$. Besides, we report the *average incremental accuracy* $\bar{\mathcal{A}}$ in the appendix. The average incremental accuracy is computed as follows:

$$\bar{\mathcal{A}} = \frac{1}{T}\sum_{t=1}^{T} \mathcal{A}_t \tag{2}$$

It indicates the average of the average accuracy over all incremental steps.

We note that the probing accuracy is calculated as the average of the test accuracy on *all* T tasks:

$$\mathcal{A}_{prob} = \frac{1}{T}\sum_{i=1}^{T} a_{prob,i}, \tag{3}$$

$a_{prob,i}$ represents the accuracy evaluated on the test set of task i after training probing classifiers. The probing accuracy is the performance when the classifier is optimal. According to the probing

Table 5: The statistics on eight datasets for incremental learning. Granularity: the classification granularity. For example, named entity recognition models classify each word into an entity type or non-entity, while intent classification models classify sentences into intent categories. # base classes: the number of classes to learn in the first task; # Inc. classes: the number of classes to learn in the incremental task (i.e., the second and subsequent tasks).

Granularity	Task	Dataset	# Classes	# Tasks	# Base Classes	# Inc. Classes	# Training Instances	# Test Instances
Sentence Level	Text Classification	Topic3Datasets	25	5	5	5	75000	46000
	Intent Classification	Clinic150	150	15	10	10	15000	4500
		Banking77	77	7	11	11	7191	2800
	Relation Extraction	FewRel	80	8	10	10	33600	11200
		TACRED	40	8	5	5	5909	1259
Word Level	Named Entity Recognition	Few-NERD	66	11	6	6	131758	230025
		Ontonotes5	18	6	8	2	59922	23836
		I2B2	16	5	8	2	59376	41397

Table 6: The details of the 9 backbones. †: Non-embedding parameters according to Biderman et al. (2023).

Architecture	Model Class	Pretrained Weights	Parameters	Layers	Hidden Dim	Link
Encoder-Only	BERT	bert-base-cased	109M	12	768	Link
		bert-large-cased	335M	24	1024	Link
Decoder-Only	GPT-NeoX	Pythia-70m	19M†	6	512	Link
		Pythia-160m	85M†	12	768	Link
		Pythia-410m	302M†	24	1024	Link
		Pythia-1b	805M†	16	2048	Link
		Pythia-1.4b	1.21B†	24	2048	Link
	GPT2	gpt2-base	124M	12	768	Link
		gpt2-large	774M	36	1280	Link

experiments in Sec. 3, the PLMs almost do not forget. Therefore, the probing performance can be regarded as the upper bound performance when using PLMs for IL.

B Datasets

The statistics of the eight datasets are summarized in Table 5. For text classification, we construct Topic3Datasets from AGNews, DBPedia, and YaHoo (Zhang et al., 2015). We remove *Sports*, *Business & Finance*,*Science & Mathematics* from YaHoo since they overlap with the classes in AGNews. We subsample 3000 training samples and 2000 test samples for each class (only 1900 test samples for the four classes in AGNews). The class order is obtained by sorting class names alphabetically and shuffling them using the random seed 1.

We use the default class order for Clinc150, Banking77, FewRel, and TACRED. We follow (Shao et al., 2023) to convert the class name to semantic labels for generative backbones. For example, in TARCRED, *org:founded_by* is converted to *organization related: founded by*.

For Few-NERD, Ontontes5, and I2B2, the class order is obtained by sorting class names alphabetically. We use the BIO schema for tagging. The *# class* represents the number of entities in Table 5.

For sentence-level classification tasks, we report the accuracy. For word-level classification tasks, we report the macro-f1 due to the class imbalance.

C Backbones

The statistics of the 9 backbones are summarised in Table 6. We download the pre-trained weights from Huggingface (Wolf et al., 2019).

C.1 Discriminant Backbones

For discriminant backbones (i.e., encoder-only backbones), we use the [CLS] feature for sentence-level classification tasks and the feature of last hidden states for word-level classification tasks. We do not use prompts for discriminant backbones. When learning each new task, we add a linear layer on top of the backbone for classification. For example, in the bert-base case, we add a linear layer whose input dimension and output dimension are 768 and 10 for learning the first task in Clinic150. Then, we add another linear layer with the same architec-

ture as the previous one when learning the second task. In the CIL scenario, the output logits over all learned categories are obtained by concatenating the logits from all classifiers. In the TIL scenario, the output logits are the logits from the classifier with the same task ID as the input sample.

C.2 Generative Backbones

For generative backbones (i.e., decoder-only backbones), we train the model to output the class name with causal language modelling loss. We do not use generative backbones for word-level classification tasks because it requires special design on input and output format (Zhao et al., 2022) and the evaluation is different from that of sequential labelling models (Monaikul et al., 2021). For text and intent classification, we use the following prompt: "Input sentence: {text}\n The label: {label}{eos token}". For relation extraction, we use the following prompt: "Input sentence: {text}\n The relationship between {head entity} and {tail entity} is {label}{eos token}". We note that we use the same prompt for all baseline models unless they have special designs on prompts. Following Sun et al. (2019), only the causal language modelling loss of "{label}{eos token}" is optimized. For the probing study, we use the last hidden states of the last word as the feature.

The max sequence length is 256 in Topic3Datasets, 50 in Clinic150, 64 in Banking77, 100 in FewRel, and 128 in other datasets. We use exactly the same backbone for all baselines unless they have special designs on backbones.

D Revisiting IL with Probing Study

D.1 Four Probing Metrics

Linear Probing trains a new linear layer on top of the backbone model. We do not use bias for linear probing classifiers because there is no significant difference between the probing performance. We train the linear probing classifier for 20 epochs with an initial learning rate of 0.001. The linear probing classifier is trained on the training data from all T tasks jointly using cross-entropy loss. The Adam (Kingma and Ba, 2014) optimizer is used, and the batch size is set as 128. We note that the training data from all tasks is mixed for optimization. Otherwise, the probing performance is degraded significantly.

Cosine Linear Probing is the same as linear probing except that the cosine similarity is adopted for calculating logits. We use the same training process as the linear probing classifier to train the cosine linear probing classifier. Hou et al. (2019) propose to use cosine linear layers for IL to avoid prediction bias towards new classes. However, we find that using cosine linear layers does not improve probing performance.

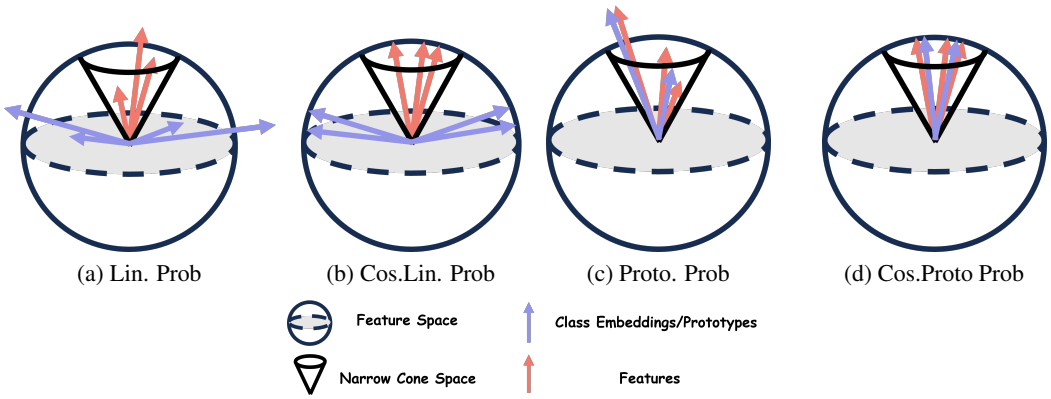
Prototype Probing requires no training of probing classifiers. It calculates the class feature centre for each class using all training data. It makes predictions of a test sample according to its Euclidean distances to all class centres. The prototype probing classifier can be regarded as a linear classifier with a weight matrix specified as all class centres.

Cosine Prototype Probing is the same as prototype probing except that the cosine similarity is adopted for calculating logits. The idea of using prototypes is widely adopted in IL (Ma et al., 2023; Cui et al., 2021; Han et al., 2020). However, we reveal that using prototypes for classification may not be the best option.

D.2 Probing Performance with Different Backbones

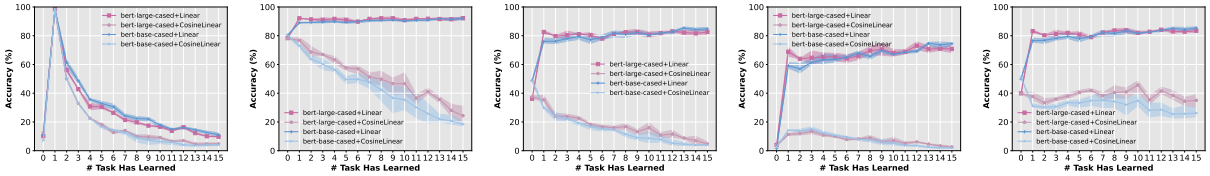
We provide the probing performance of SEQ on class-incremental intent classification in Figure 3 and 8, relation extraction in Figure 9 and 10, text classification in Figure 13 and 14, and named entity recognition in Figure 12 and 11. We summarize the findings as follows:

- The linear probing performance is significantly higher than the other three metrics across backbones, tasks, and datasets.
- For all probing metrics, the probing performance always increases when learning the first task.
- For generative backbones, the probing performance has not decreased or even increased since the second task. It indicates that the smaller backbone can adapt to downstream tasks by SEQ. The larger backbone can adapt to downstream tasks without training and maintain the knowledge during SEQ.
- For discriminant backbones, using the linear classifier maintains the probing performance, while the cosine linear classifier degrades the probing performance significantly. There is no significant difference between bert-base-cased and bert-large-cased in the probing performance.



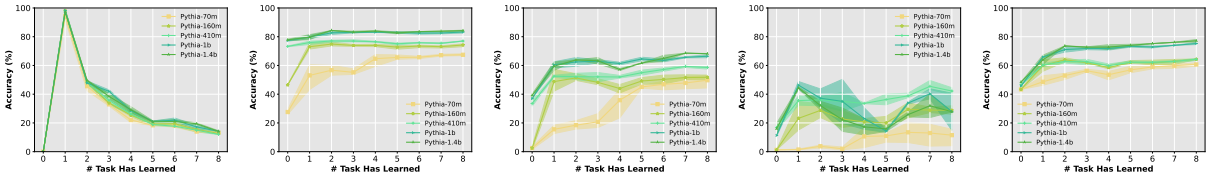
(a) Lin. Prob (b) Cos.Lin. Prob (c) Proto. Prob (d) Cos.Proto Prob

Figure 7: The illustration of different metrics for probing study.



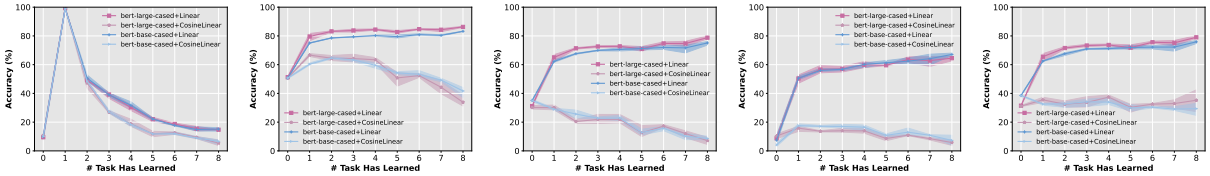
(a) Observed Performance (b) Lin. Prob (c) Cos.Lin. Prob (d) Proto. Prob (e) Cos.Proto. Prob

Figure 8: The observed and probing performance on Class-Incremental Intent Classification. The dataset is Clinic150. The backbones are generative models. (a) shows the observed performance during IL training. (b)(c)(d)(e) show the probing performance when different metrics, including linear probing, cosine linear probing, prototype probing, and cosine prototype probing.



(a) Observed Performance (b) Lin. Prob (c) Cos.Lin. Prob (d) Proto. Prob (e) Cos.Proto. Prob

Figure 9: The observed and probing performance on Class-Incremental Relation Extraction. The dataset is FewRel. The backbones are generative models. Other settings are the same as Figure 8.

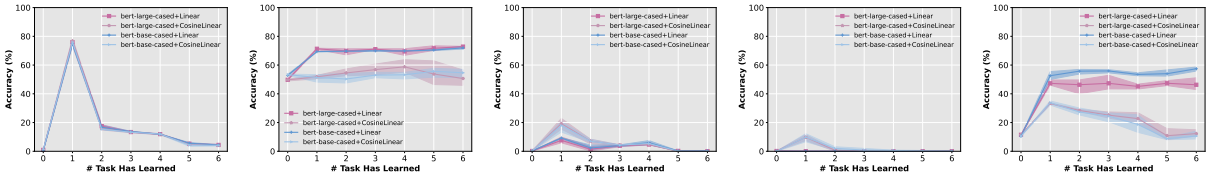


(a) Observed Performance (b) Lin. Prob (c) Cos.Lin. Prob (d) Proto. Prob (e) Cos.Proto. Prob

Figure 10: The observed and probing performance on class-incremental relation extraction. The dataset is FewRel. The backbones are discriminant models. Other settings are the same as Figure 8.

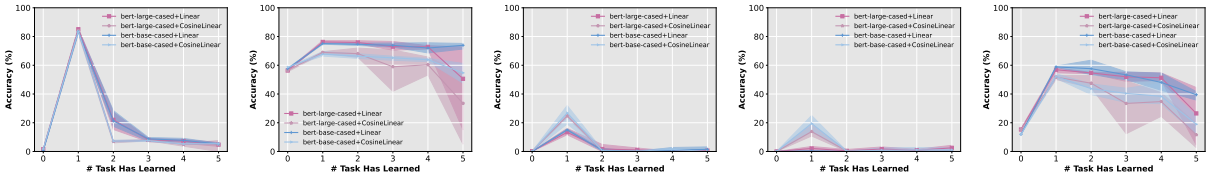
D.3 A Closer Look at Features, Word Embeddings, and Class Embeddings

To investigate the significant difference between the four probing metrics, we plot the histogram of the features, the (output) word embeddings, and the class embeddings in (cosine) linear classifiers. In this research, the features of PLMs refer to the last hidden states of the last word in generative backbones and the [CLS] token in discriminant



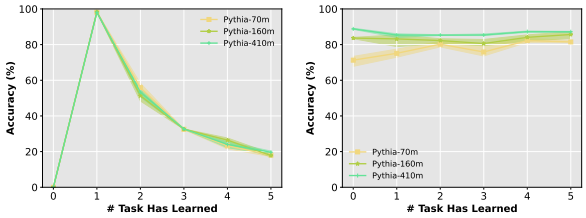
(a) Observed Performance (b) Lin. Prob (c) Cos.Lin. Prob (d) Proto. Prob (e) Cos.Proto. Prob

Figure 11: The observed and probing performance on class-incremental named entity recognition. The dataset is Ontonotes5. The backbones are discriminant models. Other settings are the same as Figure 8.



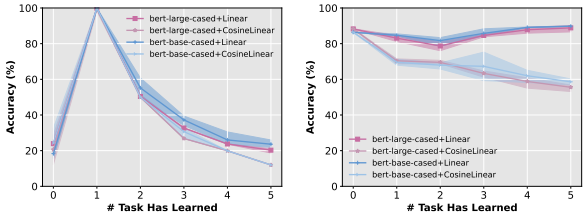
(a) Observed Performance (b) Lin. Prob (c) Cos.Lin. Prob (d) Proto. Prob (e) Cos.Proto. Prob

Figure 12: The observed and probing performance on class-incremental named entity recognition. The dataset is I2B2. The backbones are discriminant models. Other settings are the same as Figure 8.



(a) Observed Performance (b) Lin. Prob

Figure 13: The observed and probing performance on class-incremental text classification. The dataset is Topic3datasets. The backbones are generative models. Other settings are the same as Figure 8.



(a) Observed Performance (b) Lin. Prob

Figure 14: The observed and probing performance on class-incremental text classification. The dataset is Topic3datasets. The backbones are discriminant models. Other settings are the same as Figure 8.

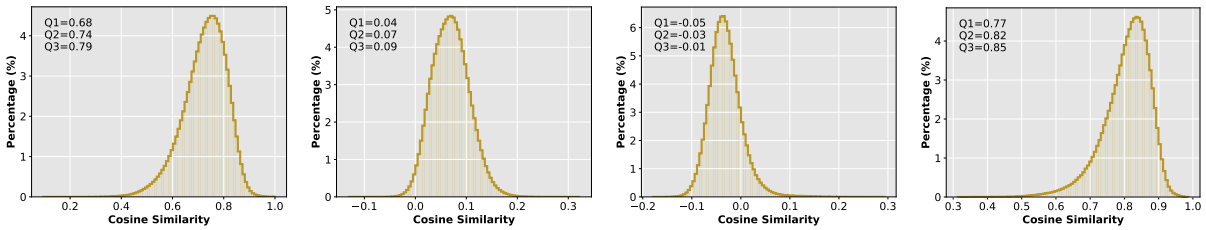
backbones. The class prototype refers to the class feature centres. The class embeddings refer to the row vectors of the weight matrix in linear layers. For example, for a linear layer whose input and output dimensions are 768 and 20, respectively, its weight matrix has the shape 20×768. Then, each row vector corresponds to a certain category, and its shape is 1×768. When using linear classifiers for prediction, the logits of a certain category are computed as the dot product between the feature and the class embeddings of that category.

During pre-training, the logits over vocabulary are computed as the dot product between features and word embeddings. In the probing study, the logits over categories are computed as the dot product between features and class embeddings. Therefore, we need to figure out the relationship between features, word embeddings, and class embeddings. We note that there is a dense layer (i.e., linear layer) between backbones and linear classifiers in BERT, and we ignore it for simplicity.

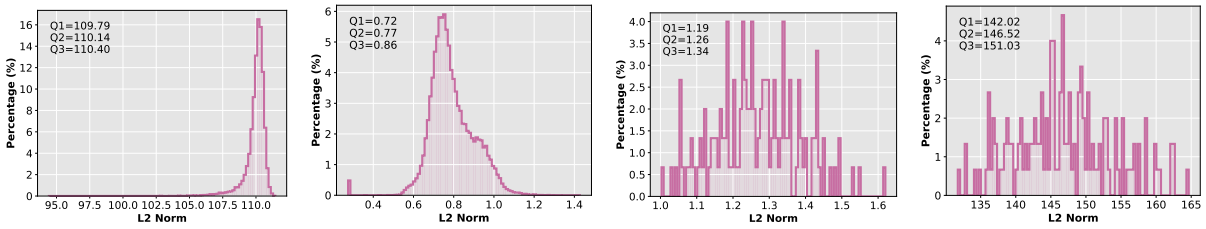
The histogram of the cosine similarity and L2 norm is provided in Figure 15 (Pythia-410m), 16 (Pythia-160m), 17 (bert-large-cased), 18 (bert-base-cased). The dataset is Clinic150. We note that the PLMs are loaded directly without fine-tuning. The features are computed on the whole training set of Clinic150. The word embeddings are loaded directly from PLMs. The class embeddings of the linear probing classifiers are obtained by training probing classifiers.

From the result, we have the following findings:

- The features of PLMs have high cosine similarity, indicating that they fall in a cone space.

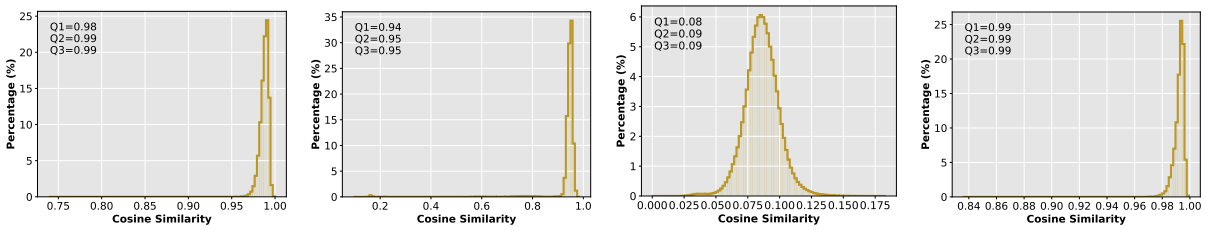


(a) Feature Sim (b) Feat-WordEmbed Sim (c) Feat-ClassEmbed Sim (d) Feat-Prototype Sim

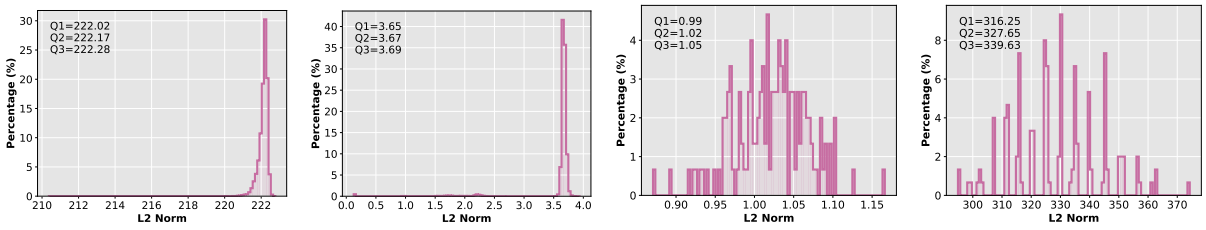


(e) Feature Norm (f) WordEmbed Norm (g) ClassEmbed Norm (h) Prototype Norm

Figure 15: The histogram of features and different embeddings of Pythia-410m. The features are calculated on the training set of Clinic150, and the output word embeddings are loaded from pre-trained weights. The class embeddings refer to the weight in the probing classifier on Clinic150. The class prototypes refer to the class feature centres on the training set of Clinic150.



(a) Feature Sim (b) Feat-WordEmbed Sim (c) Feat-ClassEmbed Sim (d) Feat-Prototype Sim

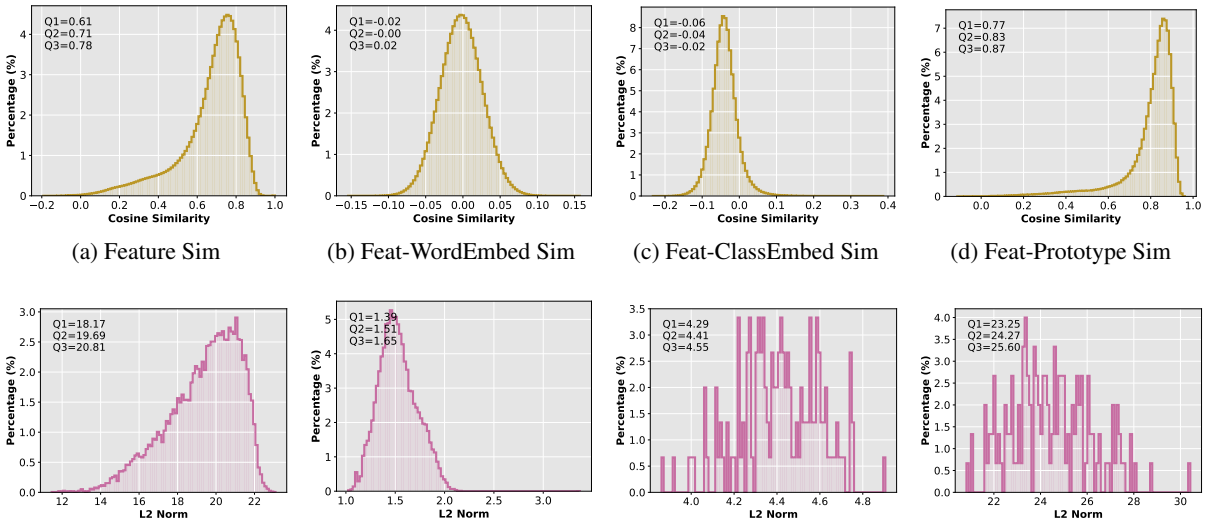


(e) Feature Norm (f) WordEmbed Norm (g) ClassEmbed Norm (h) Prototype Norm

Figure 16: The histogram of features and different embeddings of Pythia-160m. Other settings are the same as Figure 15

- The features are almost orthogonal to the word embeddings in all backbones except for Pythia-160m.
- The features are almost orthogonal to the class embeddings in all backbones.
- The features have high cosine similarity with the prototypes.
- The L2 norm of word embeddings, class embeddings have large discrepancy for all backbones except for Pythia-160m.

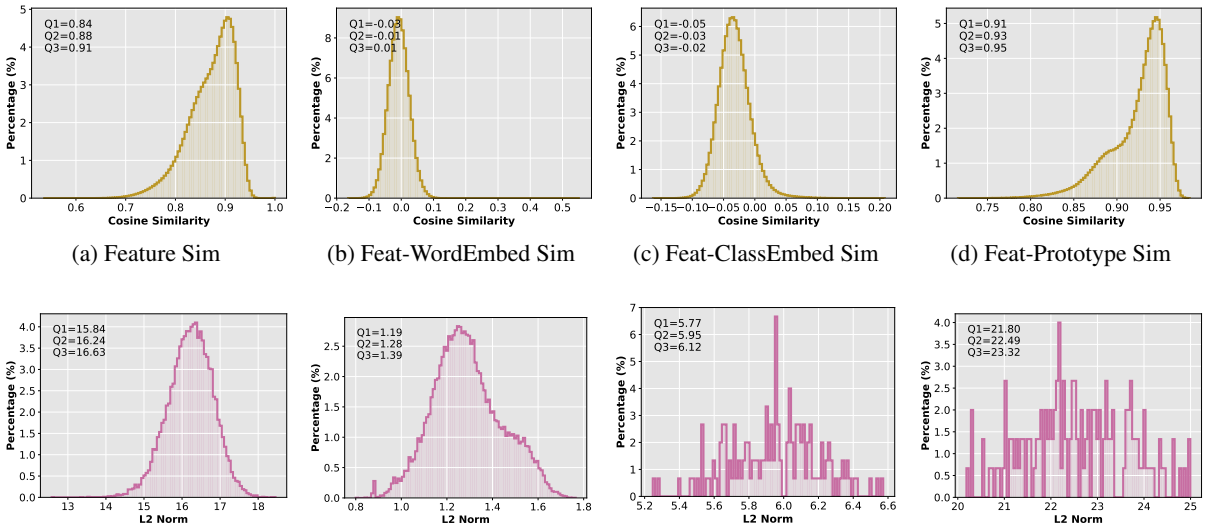
These findings explain why the linear classifier is the best option to utilize the backbone's knowledge. Specifically, the word embedding layer is also a linear classifier for pre-training, and thus, the features are most discriminative when multiplied with the class embeddings of linear classifiers. Furthermore, the discrepancy of the L2 norm of word and class embeddings suggests that the norm contains the



(a) Feature Sim (b) Feat-WordEmbed Sim (c) Feat-ClassEmbed Sim (d) Feat-Prototype Sim

(e) Feature Norm (f) WordEmbed Norm (g) ClassEmbed Norm (h) Prototype Norm

Figure 17: The histogram of features and different embeddings of bert-large-cased. Other settings are the same as Figure 15.



(a) Feature Sim (b) Feat-WordEmbed Sim (c) Feat-ClassEmbed Sim (d) Feat-Prototype Sim

(e) Feature Norm (f) WordEmbed Norm (g) ClassEmbed Norm (h) Prototype Norm

Figure 18: The histogram of features and different embeddings of bert-base-cased. Other settings are the same as Figure 15.

prior knowledge obtained from pre-training. We illustrate the four probing metrics in Figure 7.

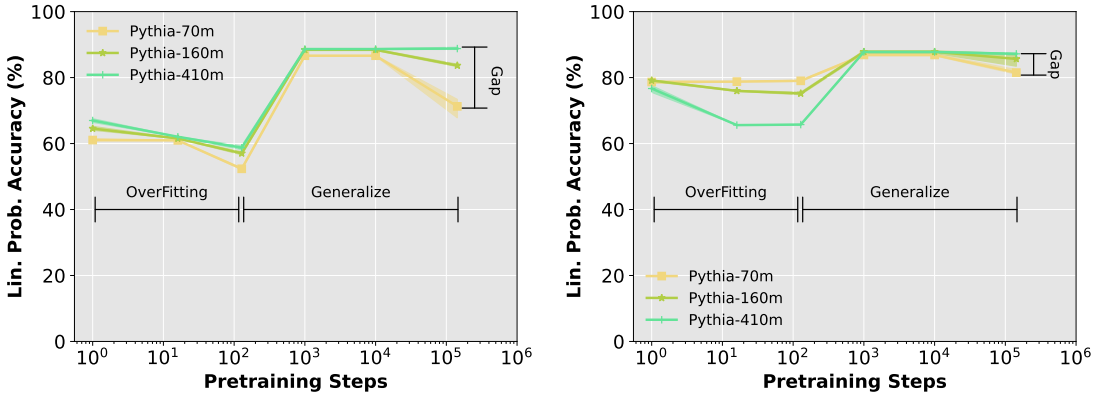
The word embeddings in Pythia-160m have high cosine similarity with features. We speculate the reason is that the parameters are not enough for generalization with causal language modelling loss.

E The Role of Pre-training

The result of text classification is shown in Figure 19. It shows a similar trend as intent classification and relation extraction in Figure 5.

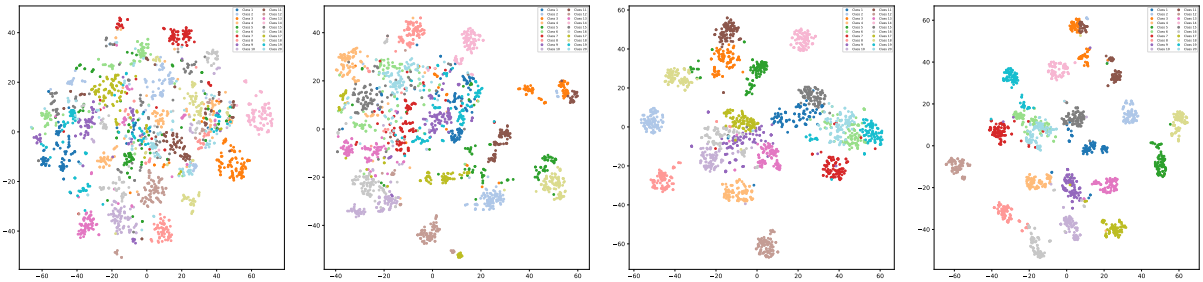
We further analyze why even a randomly-initialized model (step 0) achieves high probing performance. We use t-SNE (Van der Maaten and Hinton, 2008) to visualize the features of randomly initialized models and PLMs in Figure 20 and 21.

Figure 20a shows that randomly initialized models extract discriminative features with the Transformer architecture, which is overlooked in previous IL studies. With SEQ or pre-training, the representation ability is enhanced, and the category boundaries become clearer. For the harder dataset, FewRel, the randomly-initialized model also learns



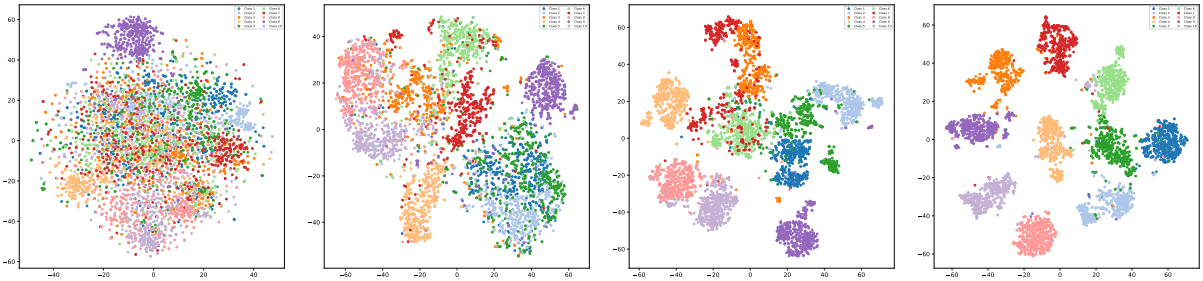
(a) TC+Before SEQ (b) TC+After SEQ

Figure 19: The linear probing performance on PLMs with different pre-training steps. (a) and (b) are evaluated before and after incremental learning using SEQ. "TC" represents that the model is evaluated on the Class-Incremental Text Classification.



(a) Before SEQ w/o pre-training (b) After SEQ w/o pre-training (c) Before SEQ w/ pre-training (d) After SEQ w/ pre-training

Figure 20: The t-SNE visualization of features on Clinic150. The backbone model is Pythia-410m. Only the first 20 classes are visualized for clarity. (a)(b): the backbone model is randomly initialized without pre-training; (c)(d): the backbone model is pre-trained.

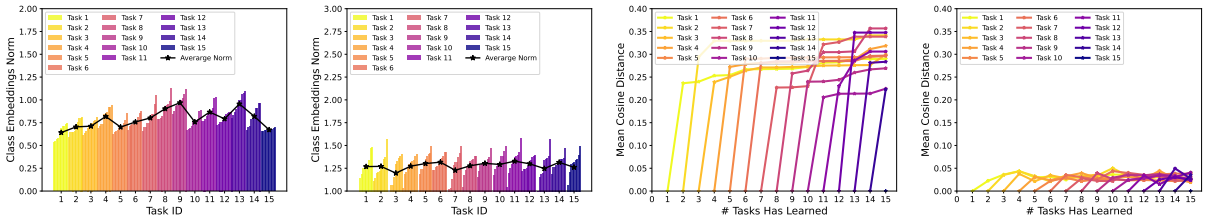


(a) Before SEQ w/o pre-training (b) After SEQ w/o pre-training (c) Before SEQ w/ pre-training (d) After SEQ w/ pre-training

Figure 21: The t-SNE visualization of features on FewRel. The backbone model is Pythia-410m. Only the first 10 classes are visualized for clarity. (a)(b): the backbone model is randomly initialized without pre-training; (c)(d) the backbone model is pre-trained.

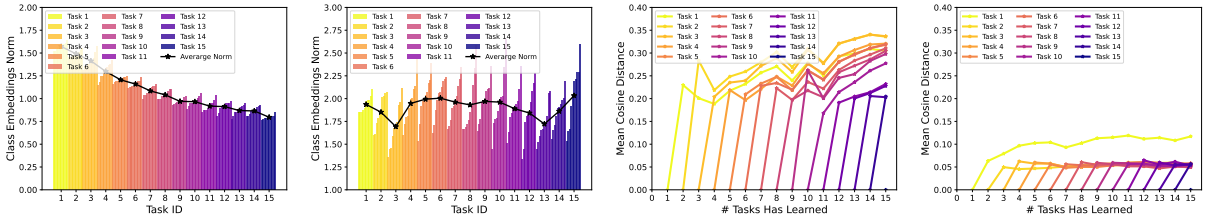
the downstream knowledge through SEQ (Figure 21a vs 21b). It explains why the performance is improved by SEQ without pre-training.

In summary, we highlight that both pre-training and the architecture of Transformers are the key factors of the anti-forgetting ability of PLMs. Most existing studies (Scialom et al., 2022; Peng et al., 2023) only attribute it to the pre-training stage. Our findings are consistent with the recent advance in the incremental learning dynamics of Transformers (Tarzanagh et al., 2023). Tarzanagh et al. (2023) discover that the rank of attention head weights



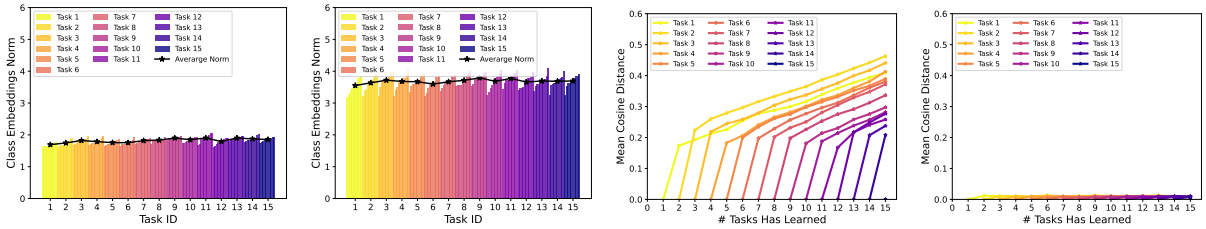
(a) Observed Classifier (b) Probing Classifier (c) Observed Classifier (d) Probing Classifier

Figure 22: Comparison between the observed and probing classifiers after SEQ on class-incremental intent classification. The backbone is Pythia-410 m and *frozen* during IL. (a)(b) show the average norm of the class embeddings of each task; (c)(d) show the average moving distance of the class embeddings of each task.



(a) Observed Classifier (b) Probing Classifier (c) Observed Classifier (d) Probing Classifier

Figure 23: Comparison between the observed and probing classifiers after SEQ on class-incremental intent classification. The backbone is bert-large-cased. (a)(b) show the average norm of the class embeddings of each task; (c)(d) show the average moving distance of the class embeddings of each task.



(a) Observed Classifier (b) Probing Classifier (c) Observed Classifier (d) Probing Classifier

Figure 24: Comparison between the observed and probing classifiers after SEQ on class-incremental intent classification. The backbone is bert-large-cased and *frozen* during IL. (a)(b) show the average norm of the class embeddings of each task; (c)(d) show the average moving distance of the class embeddings of each task.

gradually increases during training. We leave a deeper analysis of the incremental learning dynamics of the Transformers architecture to the future.

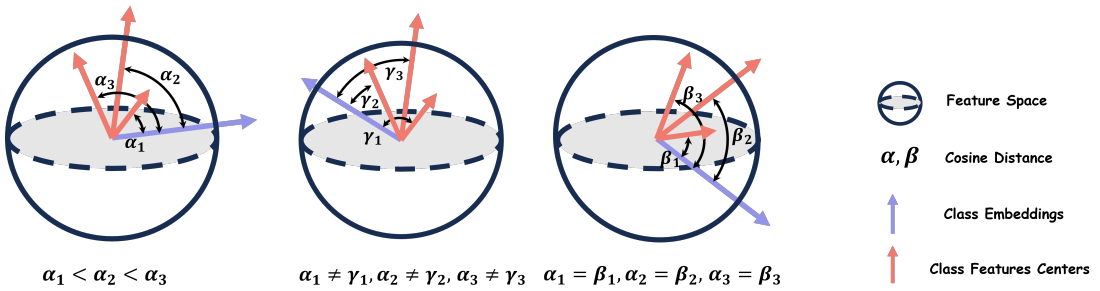
F The Forgetting in Classifiers

F.1 Overview of the Forgetting in Classifiers

Recall that we train probing classifiers on top of PLMs and achieve superior performance during SEQ. In contrast, the observed performance has dropped consistently since the second task. In previous studies (Wu et al., 2019; Hou et al., 2019), they attributed the reason for catastrophic forgetting to the bias prediction between new and old categories. Indeed, a model trained with SEQ consistently achieves high performance on the new task while the accuracy on old tasks becomes nearly zero. In other words, the model predicts larger logits on new classes and smaller logits on old classes. Then, we investigate the dynamics of L2 norm and cosine similarity between features and class embeddings during SEQ.

F.2 The Bias in the L2 Norm of Class Embeddings

We summarize the results of SEQ when the backbone is Pythia-410m in Figure 6, frozen Pythia-410m in Figure 22, bert-large-cased in Figure 23, and frozen bert-large-cased in Figure 24. The class

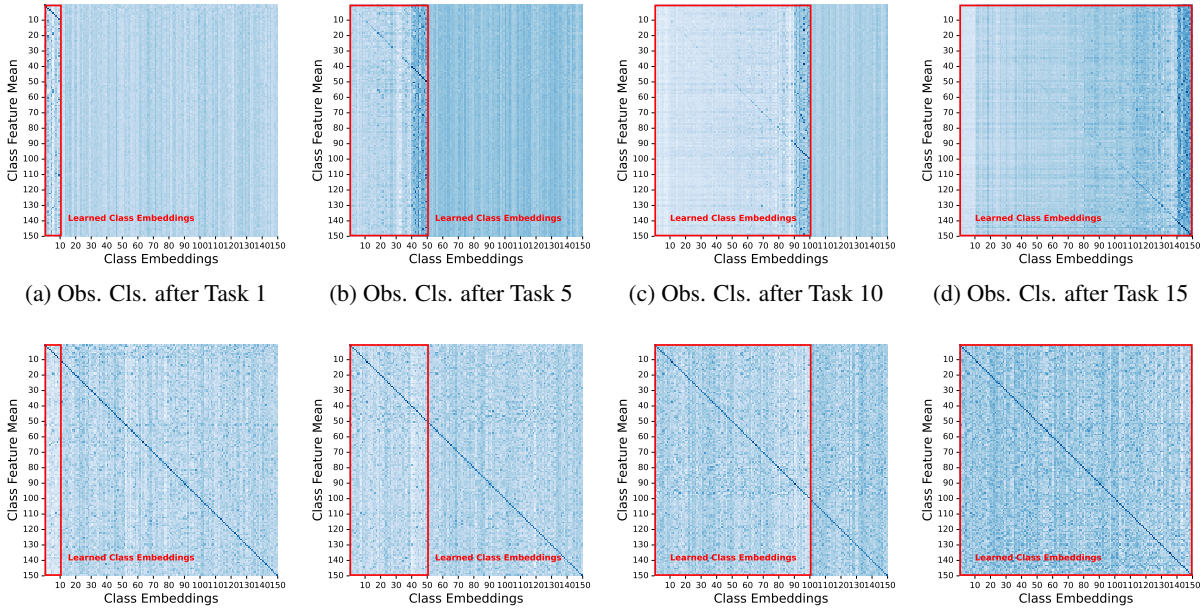


$\alpha_1 < \alpha_2 < \alpha_3$ (a) Initial Learned

$\alpha_1 \neq \gamma_1, \alpha_2 \neq \gamma_2, \alpha_3 \neq \gamma_3$ (b) Forgetting

$\alpha_1 = \beta_1, \alpha_2 = \beta_2, \alpha_3 = \beta_3$ (c) No Forgetting

Figure 25: The illustration of the moving distance of class embeddings. (a) shows the cosine similarity between class embeddings and class feature centres after a new task is learned; (b) shows that forgetting happens when the relative cosine similarity is changed; (c) shows that forgetting will not happen when the relative cosine similarity is maintained.



(a) Obs. Cls. after Task 1 (b) Obs. Cls. after Task 5 (c) Obs. Cls. after Task 10 (d) Obs. Cls. after Task 15

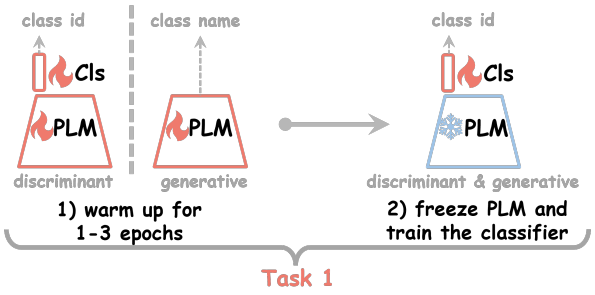
(e) Prob. Cls. after Task 1 (f) Prob. Cls. after Task 5 (g) Prob. Cls. after Task 10 (h) Prob. Cls. after Task 15

Figure 26: Comparison between the cosine similarity matrix between the linear classifiers (Obs. Cls.) and probing classifiers (Prob. Cls.). The backbone is Pythia-410m. The model is trained on class-incremental intent classification and is evaluated after learning 1,5,10,15 tasks. The result shows that the class embeddings in observed classifiers change significantly compared with probing classifiers.

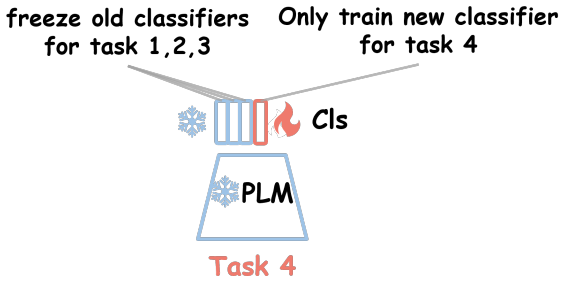
embeddings in each task are sorted according to the norm for clarity. The results show no apparent tendency in the norm of the class embeddings in both observed classifiers and probing classifiers. The norm of the class embeddings of newer classes even decreases when the backbone is bert-large-cased. Therefore, the norm is not the critical factor in the bias prediction towards new classes in SEQ.

F.3 The Bias in the Cosine Similarity between Features and Class Embeddings

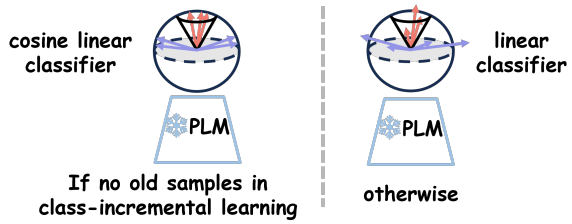
First, we define the *moving distance* to measure how the cosine similarity between features and class embeddings changes during SEQ. After a new task is learned, the relative position between features and class embeddings is optimal for this task. Intuitively, the forgetting happens only when the relative position changes. Ideally, if all pairs of features and class embeddings remain at the same angle, forgetting will not occur. In practice, we measure the cosine similarity between class em-



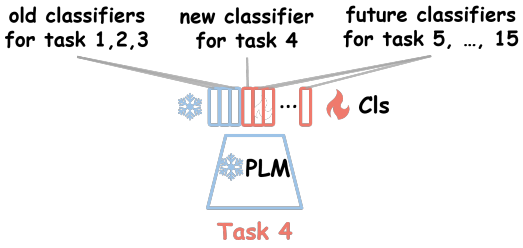
(a) S1: warm-up and freeze PLM



(b) S2: freeze old classifiers



(c) S3: use proper classifiers



(d) S4, optional: pre-allocate future classifiers

Figure 27: An illustration of the proposed SEQ*.

beddings and class feature centres instead of all features. An illustration is provided in Figure 25.

Formally, we define the moving distance of class embeddings of the i-th task at the $i+k$-th task as follows:

$$MD^i_{i+k} = \frac{1}{mn} \sum_{m=1}^{m=|\mathcal{Y}_{all}|} \sum_{n=1}^{n=|\mathcal{Y}_t|} |\mathbf{C}^i_{i+k}[m,n] - \mathbf{C}^i_i[m,n]| \tag{4}$$

The $\mathcal{Y}_{all}$ and $\mathcal{Y}_t$ represent the label set of all T tasks and the t-th task respectively. $\mathbf{C}$ represent the cosine similarity matrix between all pairs of class embeddings and class feature centres. $\mathbf{C}^i_{i+k}$ represents the cosine similarity matrix between all pairs of class embeddings from task i and all class feature centres, and it is measured at task $i+k$. $\mathbf{C}^i_{i+k}[m,n]$ is the entry of the cosine similarity matrix $\mathbf{C}^i_{i+k}$ at position $[m,n]$. This entry represents the cosine similarity between the n-th class embeddings of task t and the m-th class embeddings of all T tasks.

We summarize the moving distance of SEQ when the backbone is Pythia-410m in Figure 6, frozen Pythia-410m in Figure 22, bert-large-cased in Figure 23, and frozen bert-large-cased in Figure 24. The results clearly show that the old class embeddings in observed classifiers have drifted out of position since they were learned. Besides, the moving distance becomes larger when the backbone is frozen. In contrast, the old class embeddings in the probing classifier do not deviate much from the initial position.

We provide instances of the cosine similarity matrix in Figure 26. The cosine similarity matrix of observed classifiers changes significantly when new tasks are learned. The forgetting is more apparent when we focus on the diagonal.

In summary, we reveal that the change of relative position between class embeddings and features leads to the forgetting of classifiers. Additionally, the old knowledge is preserved if the relative position is maintained, as in probing classifiers.

G SEQ*: Boosting Performance of SEQ

We illustrate the proposed four strategies in Figure 27. **(S1) Freeze the PLMs after warm-up.** For generative backbones, we train models with causal language modelling loss to generate the ground-truth class name in the warm-up stage. For discriminant backbones, we train models with classifiers to predict the class ID with cross-entropy loss. We warm up 1 epoch for discriminant backbones and 3 epochs for generative backbones. After the warm-up stage at the first task, all parameters of backbones are frozen and will not be updated in the subsequent incremental tasks.

(S2) Freeze the old classifiers. When the model finishes training on the current task, we freeze the

classifier of the current task. When learning more tasks, we only update the new classifier while the old classifiers are frozen.

(S3) Use cosine linear classifiers only when no old data is available in the CIL scenario. Otherwise, use linear classifiers. We use linear classifiers even without old samples for named entity classification because the non-entity tokens (i.e., "Other" tokens) can be regarded as old samples from previous tasks (Zheng et al., 2022).

(S4, optional) Pre-allocate future classifiers. For example, when learning the 10 classes in the first task of Clinic150, the future classifiers from task 2 to task 15 are pre-allocated. The future classifiers are trained in advance in the first task through the softmax layer even when there are no instances from task 2 to task 15. This strategy is especially effective for generative backbones. The reason may be that it enhances the forward compatibility of classifiers, and thus, new classifiers are easier to adapt to new classes when old classifiers are frozen.

H Introduction of Baselines

Except for Topic3Datasets, we train all baselines for 5 epochs for each incremental task. On Topic3Datasets, we train all baselines for 3 epochs for each incremental task. The learning rate of backbones and classifiers are 1×10^{-5} and 1×10^{-3} respectively. We use AdamW (Loshchilov and Hutter, 2018) optimizer. We use RTX 3090 GPUs for our experiments. Each experiment is repeated three times, and the average and standard deviations are reported. We train SEQ* and all baselines with exactly the same training settings for fair comparison. We search the best hyper-parameters for each compared method and use the same hyper-parameters across backbones and datasets.

The introduction of SOTA methods is as follows:

- SEQ: SEQ refers to sequential fine-tuning. For generative backbones, SEQ trains models to output the class name with causal language modelling loss. For discriminant backbones, SEQ trains models with classifiers to predict the correct class ID.

- LAMOL (Sun et al., 2019): LAMOL trains a generative model with question-answering and generation targets and generates pseudo samples before learning each new task. The weight of the generation target $\lambda = 0.25$. The ratio of pseudo samples with respect to the training data of new task $\gamma = 0.20$. The top-k sampling for generating pseudo samples $K = 20$. There are two variations of LAMOL, including LAMOL_t and LAMOL_g. The difference is whether or not to use task-specific tokens for generation.

- L2KD (Chuang et al., 2020): L2KD proposes to add knowledge distillation targets based on LAMOL. We implement the word-level (Word-KD) variation since it performs best on text classification tasks.

- LAMOL_KD: LAMOL_KD further utilizes knowledge distillation based on LAMOL_t. Unlike L2KD, the teacher model in LAMOL_KD is trained on all previous tasks. The new data is for learning both LOAMOL targets, and the pseudo data is for word-level knowledge distillation as regularization terms. LAMOL_KD is an extension to LAMOL.

- PCLL (Zhao et al., 2022): PCLL is build upon LAMOL. Furthermore, PCLL utilizes the targets of varational autoencoders and word-level knowledge distillation to train generative models.

- LFPT5 (Qin and Joty, 2021): LFPT5 further utilizes knowledge distillation based on LAMOL. Besides, LFPT5 utilizes prompt tuning instead of fine-tuning the whole model. The number of tokens for prompt tuning is 10.

- AdapterCL (Madotto et al., 2021): AdapterCL learns one adapter (Houlsby et al., 2019) for each task. During inference in CIL, the model predicts the task ID according to causal language modelling loss, which requires T forward passes for each instance.

- LoRA (Hu et al., 2021): LoRA learns one LoRA adapter for each task. We only compare LoRA in the TIL setting. We set the rank $r = 4$ and the scaling parameter $\alpha = 8$. We use the implementation of LoRA in the PEFT library (Mangrulkar et al., 2022).

- ProgPrompt (Razdaibiedina et al., 2023): Progressive Prompt learns soft prompt for each task progressively. Following (Razdaibiedina et al., 2023), we Use a residual two-layer MLP

to encode soft prompt, and the number of soft prompt tokens for each task is 5. ProgPrompt can only be used for the TIL setting because the task ID is required during inference.

- ER: Experience replay stores representative old samples and jointly optimizes both new and old samples when learning new tasks.
- DER++ (Buzzega et al., 2020): DER++ is based on data replay. Besides, DER++ adds an MSE loss to regularize the logits of old samples between teacher and student.
- CLSER (Arani et al., 2022): CLSER is based on DER++. Besides, CLSER additionally stores two models (i.e., fast model and slow model) for selecting teacher logits when computing the MSE loss.
- SpanKL (Zhang and Chen, 2023): SpanKL converts the named entity recognition task from entity-level annotation to Span-level annotation, using a binary cross-entropy loss function and a distillation loss, with the distillation loss coefficient λ is 1.
- OCILNER (Ma et al., 2023): OCILNER utilizes contrastive learning to adaptively detect entity clusters and utilizes two distance-based methods to label non-entity tokens. The threshold for labelling Non-entity is the median class similarity of all classes in the memory.
- ExtendNER (Monaikul et al., 2021): ExtendNER utilizes knowledge distillation to review old entity types, and the distillation loss coefficient λ is 2.
- DLD (Zhang et al., 2023c): DLD improves the knowledge distillation method in ExtendNER, dividing it into negative terms and positive terms for knowledge distillation, and the distillation loss coefficient λ is 2.
- SelfTrain (Rosenberg et al., 2005; De Lange et al., 2019): SelfTrain utilizes the teacher model to directly label current data and train it together with samples of new entity types.
- RDP (Zhang et al., 2023b): RDP utilizes the previous model to pseudo-label new data and increases self-entropy loss to improve the confidence of model prediction. The self-entropy loss coefficient λ is 0.1.
- CPFD (Zhang et al., 2023a): CPFD designed a pooled feature distillation loss and proposed a confidence-based pseudo-label annotation method. The feature distillation loss coefficient λ is 2.
- ICE (Liu and Huang, 2023): ICE includes two methods, ICE_O and ICE_PLO. These methods freeze the backbone model and the old classifiers. ICE_O combine the non-entity logits with the new task logits to obtain the output probability during training. ICE_PLO uses all previous logits and new logits during training. Unlike ICE, SEQ* additionally warm-up the PLMs and pre-allocating future classifiers. Furthermore, ICE is limited to class-incremental information extraction, while SEQ* can be applied to both sentence and word-level classification tasks.
- CFNER (Zheng et al., 2022): CFNER proposes a causal effect framework to alleviate the forgetting of old entity types and uses curriculum learning methods to reduce the impact of noisy labels. The number of matched tokens K is 3.

I Revisiting SOTA Methods

I.1 CIL with Generative Backbones for Sentence-Level Tasks

The results of SOTA methods and SEQ* with Pythia-410m, Pythia-160, gpt2-base, gpt2-large are provided in Table 7, 8, 9. We compare AdapterCL only on GPT2 because Adapter is not implemented for Pythia. From the results, we have the following findings:

SEQ* shows competitive or even superior performance on all datasets. The warm-up strategy (S1) is effective when the gap between pre-training and downstream data is large (Clinic150, Banking77, FewRel, TACRED). In contrast, it is not so effective when the gap is small (Topic3Datasets). Freezing old classifiers (S2) is more important than freezing backbones. The result also validates that the forgetting results from classifiers instead of backbone PLMs. Using cosine linear classifiers (S3) is crucial when the IL scenario is CIL, and no old samples are stored. Otherwise, models will be strongly biased towards new classes. Pre-allocating future classifiers (S4) are very effective for generative backbones and brings considerable improvements. The reason may be that training future clas-

sifiers in advance reduces the overlapping between class embeddings.

The LAMOL-based methods achieve superior performance on the topic classification datasets. It indicates that generating pseudo samples is effective when the downstream data have close data distribution with the pre-training data. However, their performance is much worse than SEQ* when more tasks are learned (Clinc150), or the gap between pre-training and downstream data is larger (FewRel, TACRED). LFPT5 has the worst performance in our settings. LFPT5 was originally designed for T5 (Raffel et al., 2020), and we find that the decoder-only models do not learn to generate pseudo samples by just learning soft prompts.

SEQ* fails to achieve superior performance on gpt2-base. We find the training loss hard to decrease when adding cosine classifiers on top of the gpt2-base. On gpt2-large, SEQ* outperforms AdapterCL significantly while updating much fewer parameters.

I.2 CIL with Discriminant Backbones for Sentence-Level Tasks

We compare SEQ* with two strong rehearsal-based methods, DER++ and CLSER. Both DER++ and CLSER store teacher models for knowledge distillation and require updating all parameters in PLMs.

From the results in Table 10 and 11, we find that both DER++ and CLSER bring considerable improvement upon ER. However, SEQ* again shows competitive or superior performance on all datasets. It indicates that knowledge distillation is effective for preserving knowledge. However, fully fine-tuning PLMs causes more forgetting. Therefore, stability is more important than plasticity when designing IL algorithms for PLMs.

Besides, we have the following minor findings: (1) Pre-allocating future classifiers with discriminant backbones is less effective than generative backbones. (2) Freezing only old classifiers achieves the best performance on Clinic150 and TACRED. It shows that freezing backbone PLMs may not be necessary for some datasets. The relative position between old class embeddings and features may still be preserved when SEQ with frozen old class embeddings. (3) Using bert-large-cased may not lead to better results than bert-base-cased. We empirically find that training with bert-large-cased is more unstable than bert-base-cased.

I.3 CIL with Discriminant Backbones for Word-Level Tasks

We evaluate SEQ* on class-incremental named entity recognition since its popularity. In class-incremental named entity recognition, the non-entity tokens ("Other" tokens) can be regarded as old samples (Zheng et al., 2022). Existing methods such as (Monaikul et al., 2021; Zheng et al., 2022) utilize the Other tokens for knowledge distillation. And Zhang et al. (2023b), Zhang et al. (2023c), Zhang et al. (2023a) also verify the effectiveness of applying knowledge distillation. However, the results in Table 12, 13 show that SEQ* outperforms them by a large margin on OntoNotes5 and I2B2. On the challenging dataset Few-NERD, SEQ* beat 7 SOTA methods for class-incremental named entity recognition.

When using bert-large-cased, the performance of CFNER, DLD, and SpanKL becomes more unstable. In contrast, SEQ* are more robust to the choice of PLMs and show consistent superior performance.

Finally, we also find that the choice of classifier is crucial for class-incremental named entity recognition. When using cosine linear classifiers, the performance is degraded significantly. Therefore, we use linear classifiers for all baselines and SEQ* for a fair comparison, although CFNER, DLD, RDP, and CPFD adopt cosine linear classifiers in their original implementation. We speculate the reason is that using cosine linear classifiers prevents models from learning the prior distribution between classes, which is crucial for class-imbalanced tasks.

The result in Table 14 shows that the linear probing performance of both SEQ* and SOTA methods increases. It indicates that all models enable PLMs to adapt to downstream tasks. Furthermore, SEQ*, SelfTrain and ExtendNER have higher linear probing performance than SEQ (Lin). It indicates that forgetting causes the degradation of linear probing performance, and it can be alleviated by freezing PLMs or knowledge distillation.

I.4 TIL for Sentence-Level Tasks

We consider LoRA and ProgPrompt for the baselines in TIL. LoRA has been widely adopted for adapting PLMs to downstream tasks. The result in Table 15 shows that SEQ* achieves competitive performance but requires much fewer new parameters to learn compared with LoRA. ProgPrompt relies heavily on the choice of PLMs and shows poor

Table 7: Comparison between SOTA methods and SEQ* on sentence-level classification tasks. The backbone is Pythia-410m. The IL scenario is CIL. No old samples are stored for all models. Other notation is the same as Table 1.

	Topic3Datasets		Clinic150		Banking77		FewRel		TACRED	
	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$
LFPT5	16.78	39.23	3.48	19.99	7.98	19.87	5.52	9.05	7.60	15.90
L2KD	58.89	72.82	22.48	51.23	47.47	75.11	37.08	57.29	20.86	41.15
LAMOL_KD	49.94	70.61	41.99	68.08	52.60	73.44	25.77	49.46	29.03	48.38
LAMOL_g	**74.45**	84.36	35.43	60.67	48.40	73.93	28.10	50.81	32.70	**49.90**
LAMOL_t	74.05	**84.84**	43.37	67.77	57.00	**78.13**	28.44	51.19	28.81	48.41
PCLL	58.83	74.18	47.09	71.72	45.33	72.25	31.00	51.93	24.50	51.59
SEQ (Lin)	19.66	45.55	12.26	28.33	14.88	34.17	13.43	35.78	12.64	32.06
SEQ (Cos)	16.89	41.68	5.97	18.27	11.10	15.62	11.40	17.76	10.08	16.75
SEQ (FixB+Cos)	17.13	42.38	6.08	18.31	10.32	12.72	7.45	19.59	9.30	15.73
SEQ (FixC+Cos)	50.96	55.69	64.28	56.54	44.93	36.40	33.48	34.74	28.90	26.61
SEQ (FixBC+Cos)	53.18	57.35	62.72	56.43	44.09	33.96	33.58	33.54	28.02	26.50
SEQ (W+FixBC+Lin)	33.41	54.47	19.06	33.90	17.79	40.93	13.68	36.14	13.65	33.78
SEQ (P+W+FixBC+Lin)	33.70	52.33	27.20	37.50	15.09	36.70	17.08	37.00	14.54	34.12
SEQ* (W+FixBC+Cos)	50.77	61.69	75.96	74.29	53.76	50.02	46.12	50.36	36.55	33.51
SEQ* (P+W+FixBC+Cos)	70.56	83.69	**84.51**	**89.43**	**67.12**	75.54	**61.99**	**73.97**	**44.34**	48.52

Table 8: Comparison between SOTA methods and SEQ* on sentence-level classification tasks. The backbone is Pythia-160m. The IL scenario is CIL. No old samples are stored for all models. Other notation is the same as Table 1.

	Topic3Datasets		Clinic150		Banking77		FewRel		TACRED	
	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$
LFPT5	11.61	34.39	0.00	5.17	2.85	8.80	0.78	4.31	5.14	15.02
L2KD	52.33	70.76	21.84	47.06	37.43	56.40	21.70	47.51	10.79	28.61
LAMOL_KD	45.66	65.92	34.05	56.15	40.39	61.96	22.35	42.96	18.08	37.07
LAMOL_g	66.09	**80.98**	32.35	54.50	44.78	**66.15**	21.33	42.18	25.93	39.24
LAMOL_t	**66.11**	80.27	32.62	58.99	45.93	64.08	21.09	44.27	19.75	40.21
PCLL	44.43	66.43	39.89	64.26	41.25	63.07	26.79	48.76	21.14	**42.53**
SEQ (Lin)	19.99	44.28	10.94	25.49	13.16	32.10	12.69	34.07	11.56	31.65
SEQ (Cos)	12.33	18.78	5.22	8.22	6.75	7.28	5.99	10.67	8.77	12.55
SEQ (FixB+Cos)	8.98	14.80	5.42	8.98	7.07	8.23	6.80	8.16	9.44	12.70
SEQ (FixC+Cos)	42.89	33.35	43.53	36.47	28.66	20.04	26.55	23.74	23.51	19.29
SEQ (FixBC+Cos)	44.57	32.20	43.11	37.67	27.01	22.47	26.80	22.87	23.96	20.29
SEQ (W+FixBC+Lin)	23.35	46.60	7.51	24.93	13.24	29.78	11.70	34.38	10.59	28.72
SEQ (P+W+FixBC+Lin)	18.45	45.66	10.02	23.56	13.73	31.39	13.54	33.05	10.90	29.25
SEQ* (W+FixBC+Cos)	42.30	47.28	67.97	62.37	45.80	37.40	47.38	50.54	34.27	31.25
SEQ* (P+W+FixBC+Cos)	57.71	52.33	**77.94**	**83.73**	**62.66**	64.81	**59.57**	**72.23**	**36.99**	40.21

performance on Pythia-410m and 160m. Since the scenario of TIL is much simpler than that of CIL, and all methods achieve high accuracy, we did not conduct as many experiments in TIL as in CIL.

Table 9: Comparison between SOTA methods and SEQ* on clinic150. The backbone is gpt2-base and gpt2-large. The IL scenario is CIL. No old samples are stored for all models. $\mathcal{A}_{prob,0}$: the linear probing performance before IL; $\mathcal{A}_{prob,T}$: the linear probing performance after IL. Other notation is the same as Table 1.

Backbone	Method	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_{prob,0}$	$\mathcal{A}_{prob,T}$	# New Params per Task
gpt2-base	**AdapterCL**	71.64	75.57	83.71	84.05	894K
	SEQ (Cos)	1.31	4.34	83.71	85.28	7.68K
	SEQ* (W+FixBC+Cos)	4.55	9.22	83.71	86.84	7.68K
	SEQ* (P+W+FixBC+Cos)	10.17	14.59	83.71	86.76	7.68K
gpt2-large	**AdapterCL**	70.42	78.91	91.95	91.87	7421K
	SEQ (Cos)	6.48	20.25	91.95	93.89	12.8K
	SEQ* (W+FixBC+Cos)	72.95	68.28	91.95	94.06	12.8K
	SEQ* (P+W+FixBC+Cos)	87.93	91.41	91.95	94.18	12.8K

Table 10: Comparison between SOTA methods and SEQ* on sentence-level classification tasks. The backbone is bert-large-cased. The IL scenario is CIL. Each model stores 1 sample for each class. Other notation is the same as Table 1.

	Topic3Datasets		Clinic150		Banking77		FewRel		TACRED	
	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$
ER	56.65	74.93	65.32	82.57	49.07	72.79	38.94	66.23	36.36	54.69
CLSER	56.36	74.68	73.12	86.80	50.07	70.85	45.59	67.9	42.53	57.41
DER++	61.49	78.83	72.7	86.49	56.15	76.67	44.41	68.32	42.5	59.05
SEQ (Lin)	20.28	45.4	7.62	25.45	13.19	22.78	14.72	36.11	13.6	33.08
SEQ (Cos)	17.55	24.17	8.33	15.56	10.04	12.4	9.55	15.8	9.46	14.79
SEQ (FixB+Lin)	44.58	67.70	24.35	57.02	20.29	49.23	14.09	35.62	12.50	26.12
SEQ (FixC+Lin)	58.24	76.63	**78.86**	**89.90**	58.34	75.17	44.36	69.77	**45.26**	**60.46**
SEQ (FixBC+Lin)	61.20	80.52	47.11	70.17	46.29	64.24	29.43	49.85	23.14	32.73
SEQ (W+FixBC+Cos)	36.52	59.05	44.9	58.49	31.09	51.25	34.15	49.52	23.8	40.78
SEQ (P+W+FixBC+Cos)	33.48	55.2	10.95	12.44	14.53	25.84	21.7	35.71	16.9	30.78
SEQ* (W+FixBC+Lin)	62.63	80.63	73.12	86.81	**62.02**	**80.04**	**51.57**	**72.5**	43.8	58.47
SEQ* (P+W+FixBC+Lin)	**64.39**	**81.92**	72.52	86.49	61.15	79.42	49.71	70.87	41.98	58.2

Table 11: Comparison between SOTA methods and SEQ* on sentence-level classification tasks. The backbone is bert-base-cased. The IL scenario is CIL. Each model stores 1 sample for each class. Other notation is the same as Table 1.

	Topic3Datasets		Clinic150		Banking77		FewRel		TACRED	
	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$
ER	51.90	72.76	63.11	82.33	49.76	73.20	40.32	64.02	35.50	53.17
CLSER	57.23	75.67	71.77	86.46	51.17	74.75	43.33	67.63	42.13	55.24
DER++	**64.78**	79.29	69.05	86.13	55.84	76.63	44.67	67.73	40.58	56.11
SEQ (Lin)	23.62	48.38	11.09	30.88	14.07	23.78	15.16	36.29	12.77	35.60
SEQ (Cos)	17.65	25.84	7.06	14.77	9.48	11.80	8.54	13.78	9.98	16.54
SEQ (FixB+Lin)	39.98	64.44	25.26	57.33	21.07	49.61	14.38	36.56	14.36	26.20
SEQ (FixC+Lin)	53.90	74.28	**78.08**	**89.03**	59.64	77.19	46.06	66.87	**45.65**	**57.79**
SEQ (FixBC+Lin)	61.81	79.63	44.97	69.10	40.77	63.96	32.24	50.07	21.26	29.28
SEQ (W+FixBC+Cos)	31.91	54.39	13.11	14.09	15.74	25.82	18.47	31.04	22.53	40.77
SEQ (P+W+FixBC+Cos)	26.45	51.71	40.23	53.29	30.73	49.88	33.96	49.64	14.86	23.29
SEQ* (W+FixBC+Lin)	63.12	**81.77**	69.30	84.21	**61.95**	80.18	**53.66**	**72.56**	40.63	56.44
SEQ* (P+W+FixBC+Lin)	63.58	80.39	68.50	83.24	61.94	**80.20**	52.76	72.41	42.08	56.95

Table 12: Comparison between SOTA methods and SEQ* on word-level classification tasks. The backbone is bert-large-cased. The IL scenario is CIL. No old samples are stored for all models. Other notation is the same as Table 1.

	Few-NERD		**OntoNotes5**		**I2B2**	
	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$
SpanKL	0.00	0.00	0.00	0.00	0.00	1.98
OCILNER	22.83	31.78	37.67	53.58	36.94	52.12
ExtendNER	20.69	31.77	46.46	57.65	26.22	44.62
DLD	21.46	38.42	48.96	58.17	0.00	28.75
SelfTrain	25.73	40.48	49.31	57.70	36.83	55.00
RDP	<u>30.66</u>	<u>45.93</u>	54.41	64.30	42.96	61.59
CPFD	**35.65**	**48.88**	59.02	64.51	24.75	52.98
ICE_O	25.61	29.35	47.19	48.21	53.03	55.07
ICE_PLO	17.54	22.11	42.53	46.16	47.76	53.21
CFNER	29.90	44.02	48.62	57.96	1.23	27.05
SEQ (Lin)	3.31	16.49	4.42	21.42	4.77	25.46
SEQ (W+FixBC+Cos)	9.21	21.40	32.60	50.51	43.64	59.50
SEQ (P+W+FixBC+Cos)	5.09	18.25	33.95	50.94	46.25	61.71
SEQ* (W+FixBC+Lin)	27.72	42.27	<u>67.21</u>	<u>73.23</u>	**74.76**	**77.17**
SEQ* (P+W+FixBC+Lin)	28.52	42.57	**68.87**	**73.70**	<u>72.66</u>	<u>75.79</u>

Table 13: Comparison between SOTA methods and SEQ* on word-level classification tasks. The backbone is bert-base-cased. The IL scenario is CIL. No old samples are stored for all models. Other notation is the same as Table 1.

	Few-NERD		**OntoNotes5**		**I2B2**	
	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	$\mathcal{A}_T$	$\bar{\mathcal{A}}$
SpanKL	18.26	39.28	40.10	53.88	6.12	34.84
OCILNER	18.44	29.80	39.99	54.70	27.27	47.74
ExtendNER	20.02	36.34	48.08	57.02	20.02	36.34
DLD	20.75	35.53	47.23	58.04	30.50	48.03
SelfTrain	23.46	39.88	51.08	57.41	23.60	39.49
RDP	27.08	43.43	50.45	60.32	40.38	58.12
CPFD	**34.65**	**46.92**	55.58	59.34	43.52	56.15
ICE_O	28.98	33.94	51.81	53.06	49.12	54.56
ICE_PLO	19.94	29.31	46.52	49.79	47.76	53.35
CFNER	27.70	42.13	58.07	63.76	35.42	51.44
SEQ (Lin)	2.97	16.21	4.38	21.20	5.26	25.26
SEQ (W+FixBC+Cos)	7.26	19.72	29.12	47.60	45.95	61.29
SEQ (P+W+FixBC+Cos)	3.17	19.65	29.70	48.30	47.10	60.42
SEQ* (W+FixBC+Lin)	28.13	42.72	66.99	71.80	71.76	73.71
SEQ* (P+W+FixBC+Lin)	28.21	43.06	**67.39**	**72.27**	**72.51**	**75.48**

Table 14: The linear probing performance with backbone bert-base-cased. Other settings are the same as Table 13.

	OntoNotes5		**I2B2**	
	Before IL	**After IL**	**Before IL**	**After IL**
SEQ (Lin)	52.89±0.41	71.93±1.04	58.18±0.92	73.40±0.94
SelfTrain		74.29±0.75		74.54±0.57
ExtendNER		73.96±0.85		76.40±0.55
SEQ* (P+W+FixBC+Lin)		74.05±1.02		75.08±1.15

Table 15: Comparison between SOTA methods and SEQ* on clinic150 and FewRel. The backbone is Pythia-410m, Pythia-160m, bert-large-cased and bert-base-cased. The IL scenario is TIL. No old samples are stored for all models. Other notation is the same as Table 1.

Backbone	Method	Clinic150			FewRel		
		$\mathcal{A}_T$	$\bar{\mathcal{A}}$	# New Params per Task	$\mathcal{A}_T$	$\bar{\mathcal{A}}$	# New Params per Task
Pythia-410m	**LoRA**	97.04	98.60	393K	95.29	96.62	393K
	ProgPrompt	57.91	98.00	5.12K	46.73	55.05	5.12K
	SEQ* (W+FixBC+Lin)	98.04	98.01	10.24K	90.02	93.03	10.24K
	SEQ* (P+W+FixBC+Lin)	98.27	98.02	10.24K	91.25	93.61	10.24K
Pythia-160m	**LoRA**	51.62	50.81	147K	83.93	87.35	147K
	ProgPrompt	13.00	27.03	3.84K	14.79	22.36	3.84K
	SEQ* (W+FixBC+Lin)	95.94	96.75	7.68K	88.42	91.35	7.68K
	SEQ* (P+W+FixBC+Lin)	96.42	97.09	7.68K	88.62	92.23	7.68K
bert-large-cased	**LoRA**	98.86	98.97	393K	93.63	95.27	393K
	ProgPrompt†	94.82	96.85	15.36K	90.96	92.90	15.36K
	SEQ* (W+FixBC+Lin)	97.80	97.80	10.24K	85.56	87.75	10.24K
	SEQ* (P+W+FixBC+Lin)	97.70	97.64	10.24K	85.93	88.29	10.24K
bert-base-cased	**LoRA**	98.53	98.46	147K	94.00	95.42	147K
	ProgPrompt†	98.15	98.25	11.52K	92.38	94.03	11.52K
	SEQ* (W+FixBC+Lin)	96.71	96.64	7.68K	86.50	88.64	7.68K
	SEQ* (P+W+FixBC+Lin)	96.30	96.20	7.68K	86.65	88.73	7.68K