Forging Multiple Training Objectives for Pre-trained Language Models via Meta-Learning

Hongqiu Wu\textsuperscript{1,2,♠}, Ruixue Ding\textsuperscript{♠}, Hai Zhao\textsuperscript{1,2,∗}, Boli Chen\textsuperscript{♠}, Pengjun Xie\textsuperscript{♠}, Fei Huang\textsuperscript{♠}, Min Zhang\textsuperscript{3}

\textsuperscript{1}Department of Computer Science and Engineering, Shanghai Jiao Tong University
\textsuperscript{2}Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai Jiao Tong University
\textsuperscript{3}School of Computer Science and Technology, Soochow University
\textsuperscript{♠}Damo Academy, Alibaba Group

wuhongqiu@sjtu.edu.cn, zhaohai@cs.sjtu.edu.cn, \{ada.drx,boli.cbl,chengchen.xpj,f.huang\}@alibaba-inc.com, minzhang@suda.edu.cn

Abstract

Multiple pre-training objectives fill the vacancy of the understanding capability of single-objective language modeling, which serves the ultimate purpose of pre-trained language models (PrLMs), generalizing well on a mass of scenarios. However, learning multiple training objectives in a single model is challenging due to the unknown relative significance as well as the potential contrariety between them. Empirical studies have shown that the current objective sampling in an ad-hoc manual setting makes the learned language representation barely converge to the desired optimum. Thus, we propose MOMETAS, a novel adaptive sampler based on meta-learning, which learns the latent sampling pattern on arbitrary pre-training objectives. Such a design is lightweight with negligible additional training overhead. To validate our approach, we adopt five objectives and conduct continual pre-training with BERT-base and BERT-large models, where MOMETAS demonstrates universal performance gain over other rule-based sampling strategies on 14 natural language processing tasks\textsuperscript{1}.

1 Introduction

It is appealing for deep neural language models to generalize well on multiple downstream tasks through large-scale language pre-training, e.g. BERT (Devlin et al., 2019), ELECTRA (Clark et al., 2020), DeBERTa (He et al., 2021) and GPT (Brown et al., 2020). Most pre-trained language models (PrLMs) rely on only one or two pre-training objectives, from Masked Language Modeling (MLM), Next Sentence Prediction (NSP) (Devlin et al., 2019), Sentence Order Prediction (SOP) (Lan et al., 2020) and Permutation Language Modeling (PLM) (Yang et al., 2019). Even though PrLMs are intended for high generalization, studies show that they are not always all-rounded and tend to be particularly weak in some aspects (Li and Zhao, 2021a; Li et al., 2020; Yang et al., 2019), while an ultimate purpose of a language understanding system is to stand for the nice initialization on a mass of scenarios simultaneously and effectively.

With the birth of more and more pre-training objectives, a number of specific ones beyond are found of great benefit to enhance task-level understanding capability, e.g. contrastive learning (Gao et al., 2021), adversarial training (Wu and Zhao, 2022), knowledge injection (Xiong et al., 2020). To enjoy the merits of all worlds and let the model generalize better on more seen or perhaps unseen tasks, there naturally comes a need to combine all these objectives in an organic manner.

However, learning multiple pre-training objectives simultaneously in a single model is challenging (Chen et al., 2018; Yu et al., 2020). A well-known issue is negative transfer (Wang et al., 2019b) in which learning well on one objective impairs another. More importantly, the relative significance between all objectives is supposed to be scheduled. For instance, NSP can take little effect on the model due to its simpleness in the mature stage of training. However, it is of great difficulty to heuristically tune such a ratio considering the large amounts of compute to pre-train once. In most cases we tentatively treat all of them equally (Liu et al., 2019; Lewis et al., 2020), which makes the learned language representation barely converge to the optimal point and limits the model performance (Chen et al., 2018; Wang et al., 2019b).

To forge multiple training objectives for PrLMs,
this paper presents to learn an optimal sampling strategy so that the more informative objective is more likely to be chosen. The backbone is meta-learning (Thrun and Pratt, 1998) and thus we call it Multi-Objective META-Sampler (MOMETAS).

In the proposed framework, we redesign the pre-training process into two phases, meta-train and meta-test. The model is trained alternately on one sampled objective at each step during meta-train, while the sampling distribution is then updated during meta-test by measuring the relative contribution of each objective. The training design is lightweight with little additional overhead to guarantee the pre-training efficiency. To validate our approach, we consider five pre-training objectives (e.g. for sentence embedding, knowledge caption, syntactic understanding) and continue to pre-train with BERT-base and BERT-large, where MOMETAS demonstrates universal performance gain over other rule-based sampling strategies on 14 natural language processing tasks.

2 Related Work

2.1 Multiple Pre-training Objectives

Our work is dedicated to improvement of learning multiple pre-training objectives on a single language model (Liu et al., 2019; Lewis et al., 2020). Language pre-training is well-studied in recent years and there are various potential objectives proposed, e.g. to enhance general language representation (Lewis et al., 2020), text generation (Yang et al., 2019; Dong et al., 2019), sentence embedding (Gao et al., 2021; Li and Zhao, 2021a), dialogue understanding (Xu and Zhao, 2021; Li and Zhao, 2021b). MOMETAS is designed to bring them together organically.

Our work is related to balancing training in multi-task networks, e.g. gradient normalization (Chen et al., 2018), projecting conflicting gradients (Yu et al., 2020), weighting training loss based on uncertainty (Kendall et al., 2018). For PrLMs, it is explored more on fine-tuning (Stickland and Murray, 2019; Raffel et al., 2020; Poth et al., 2021). In practice, BERT-style pre-training like MLM (Devlin et al., 2019) establishes self-supervised objectives through certain transformations on text data. From this point of view, our work is similar to reweighting training samples (Alain et al., 2015; Ren et al., 2018) or data selection (Schulman et al., 2016; Wang et al., 2020a).

A related application in natural language processing is to train multilingual models (Arivazhagan et al., 2019; Wang et al., 2020b,c; Zhou et al., 2021; Wang et al., 2021). For instance, MultiDDS (Wang et al., 2020b) learns a data scorer to balance the data usage of languages. However, designing pre-training is more challenging for lack of prior knowledge, e.g. data size (Johnson et al., 2017), data resource (Neubig and Hu, 2018). Besides, one can not access to real downstream tasks. All these can lead to so different optimization designs.

2.2 Meta Learning

Meta-Learning (Learning to Learn) (Thrun and Pratt, 1998) has a long history with vast contributing literature, whereas we could only mention several related works here. Ravi and Larochelle (2017) designs an LSTM-based meta-learner to learn the update rule for few shot learning. Finn et al. (2017) proposes MAML to learn an optimized initialization ready for fast adaption to new tasks. The idea also emerges in recent natural language processing, e.g. generating the text mask for MLM (Kang et al., 2020), optimizing the first-order approximation of dropout to learn dynamic attention pattern (Wu et al., 2021), leveraging MAML-inspired pre-training to find a global representation of downstream tasks (Lv et al., 2020; Ke et al., 2021).

3 Multi-Objective Meta-Sampler

In this section, we first take an overview of our meta-learning framework. What follows is the preliminaries of the pre-training setting as well as a number of ruled-based samplers. Then we discuss the details of our meta-sampler.

3.1 Overview

As depicted in Figure 1, we learn the problem in two phases, meta-train and meta-test. In meta-train, the model is trained and updated on a series of pre-training objectives sampled through MOMETAS one by one. After a number of steps, it goes through meta-test, where we evaluate the model over all objectives in one shot. The evaluation is done on a clean validation set in addition to the training one. Based on the evaluation feedback, MOMETAS is then updated. We repeat such train-test cycles until the end of pre-training.

3.2 Multi-Objective Pre-training

In our multi-objective pre-training, the model is trained on m different objectives. The input text
of each objectives passes a common encoder to obtain the shared language representation and then output through a specific layer (or head). We denote all objectives as $\{T_1, T_2, \cdots, T_m\}$, the sampling of which is subject to the latent distribution $P_D$. At each training step $t$, a single objective $T_t \in \{T_1, T_2, \cdots, T_m\}$ is sampled from $P_D$.

### 3.3 Rule-based Samplers

We first consider several rule-based samplers:

- **Uniform-based**: The most straightforward and simplest approach is to make uniform sampling over all objectives. It equals conventional multi-objective training and multi-task learning. However, when the number of objectives is up, it is hard to guarantee the training efficiency, since some simpler objectives come close to convergence early, while some more difficult ones still require a large number of steps to learn well.

- **Gradient-based**: Gradient acts as a contributing signal of the training state of a network when making gradient descent (Ravi and Larochelle, 2017; Wang et al., 2020b; Yu et al., 2020). Larger gradient may have a greater impact on updating its parameters. An intuitive idea is to sample more on those objectives with large gradients, while less on those with small gradients which tend to take minimal impacts on the network. Computationally, we may take the norm of gradients over all encoder parameters (Ravi and Larochelle, 2017).

- **Loss-based**: Similar as above, loss acts as another contributing signal of how well a certain objective is learned (Kendall et al., 2018). More specifically, we may compute the inverse training rate (IR) by dividing the current loss by its initial value, so that lower IR corresponds to a faster training rate for the objective. Thus, the idea is to sample more on those objectives with higher inverse training rates.

### 3.4 Meta-Sampler

Both gradient-based and loss-based approaches merely focus on the state of a single objective in an ad-hoc manner but do not take into account the coupling between them, which makes it hard to achieve the optimal point across all objectives.

Thus, we propose to learn a meta-sampler MOMETAS parametrized as $\psi = P_D$, based on meta-learning. Suppose that we sample a single objective at each step $t$ from $P_D$ during meta-train and obtain a sequence of objectives:

$$\tau = \{T_1, T_2, \cdots, T_K\}, \tau \sim P_D$$

where $K$ refers to the number of steps of meta-train (we call it meta length in the paper). In the following meta-test, we evaluate the model over all objectives $T_1:K$ on an additional validation set $Y$. The goal of MOMETAS is to learn well or earn more gain on all objectives, that is to maximize:

$$J(\psi) = E_{\tau \sim P_D} [R(\tau)]$$

where $R(\tau)$ refers to the overall gain given $\tau$. 

6485
Since \( J(\psi) \) is non-differentiable, it is impossible to apply normal gradient-based methods to update MOMETAS which makes sampling from different objectives. Following REINFORCE (Sutton et al., 1999), we take a number of policy gradient steps to accommodate the non-differentiable operations of sampling, that is:

\[
\psi \leftarrow \psi + \beta \sum_{t=1}^{K} \nabla_{\psi} \log P(T_t; \psi) R(\tau) \tag{2}
\]

where \( \beta \) refers to the meta step size. From this perspective, \( R(\tau) \) can be viewed as a rewarding function of training gain. Note that \( R(\tau) \) is only obtained at the end of meta-train \((t = K)\).

Meta length \( K \) indicates the accumulation of meta knowledge. Intuitively, larger \( K \) comes to more training samples until each meta update step, which stabilizes the training process but lowers down the sensitivity of MOMETAS.

### 3.4.1 Individual Rewarding

We further explore the details of the rewarding function \( R(\tau) \). We first let \( r^i \) be the individual gain on each objective \((i = 1 \sim m)\) so that \( R(\tau) = \sum_{i=1}^{m} r^i \). However, our empirical results show that simply letting \( r^i \) be the opposite of each evaluation loss merely leads to limited performance. This is caused by the problem that it cannot address the issue of negative transfer. Suppose that there is a dominant objective, trained well so that the loss of it is continually down. The real situation can be that the overall loss is declining, while the individual losses of certain objectives are still rising, even though MOMETAS is positively rewarded.

To destroy such confusion, we let \( r^i \) be the loss drop of each objective. Specifically, to compute each loss drop, we always maintain the last loss value as the baseline \( b^i \) (the evaluation loss from last meta-test). Then we compare the current loss value \( a^i \) (from current meta-test) with it. Because the magnitude of loss differs from objectives, we further compute the relative loss drop by dividing it by the baseline \( b^i \). Hence, the final rewarding function can be formulated as:

\[
R(\tau) = \sum_{i=1}^{m} \frac{b^i - a^i}{b^i} \tag{3}
\]

where \( b^i \) and \( a^i \) refer to the loss values of the last meta-test and current meta-test respectively. Such rewarding function forces MOMETAS to explore the optimal sampling pattern which is useful across all pre-training objectives.

### 3.4.2 Entropy Regularization

To further escape from the local optimum, we impose maximum entropy regularization as an additional constraint (Haarnoja et al., 2018), which is widely used in stochastic reinforcement learning. The idea behind this is that smaller entropy means more deterministic sampling from the distribution and MOMETAS will be punished in this situation, which encourages MOMETAS to explore and allows it to step out of the local optimal point. Hence, the training objective of MOMETAS comes to:

\[
J(\psi) = E_{\tau \sim P_D} \left[ R(\tau) + \lambda H(\psi) \right] \tag{4}
\]

where \( H(\psi) \) refers to the entropy regularization term. We find good performances when the temperature parameter \( \lambda \) is set to \( 1 \sim 3 \).

### 3.4.3 Algorithm

Then we present our meta-learning algorithm, which is summarized in Algorithm 1. Specifically, we first initialize MOMETAS distribution \( P_D \) with uniform distribution. In meta-train, the model is fed with \( K \) sampled pre-training objectives one by one. At each step \( t \), we need to record every single sampling \( T_t \) in order to update MOMETAS later. What follows is meta-test, where the model is evaluated on the validation set \( \mathcal{V} \). MOMETAS will be rewarded based on the evaluation feedback and then updated so as to be ready for the next meta-train. We repeat such a train-test cycle for times until model convergence. Note that we fetch the validation samples from \( \mathcal{V} \) through random sampling to guarantee the training efficiency.

---

**Algorithm 1 Pre-train with MOMETAS**

**Input:** Model \( \theta \), \( m \) pre-training objectives \( \{T^1, T^2, \ldots, T^m\} \), meta length \( K \), MOMETAS distribution \( P_D \), validation set \( \mathcal{V} \)

1. Initialize \( D \) with uniform distribution
2. while not converged do
   3. Empty \( \tau \)
   4. for \( t = 1 \) to \( K \) do
      5. Sample one objective \( T_t \sim P_D \)
      6. Update model parameters \( \theta_t \)
      7. Append \( T_t \) into \( \tau \)
   8. end for
   9. Fetch data for each objective from \( \mathcal{V} \)
10. Evaluate with model parameters \( \theta_K \)
11. Compute reward via Eq. 3
12. Update \( P_D \) via Eq. 2
13. end while
Table 1: GLUE test results under different sampling strategies. BERT\textsubscript{base} refers to the reported results in Devlin et al. (2019) while BERT\textsubscript{base (Ours)} refers to our rerun results. Due to limited number of submissions per day, we do not report the results over multiple runs in Table 1 (for multiple runs, please refer to Table 2).

<table>
<thead>
<tr>
<th>Method</th>
<th>CoLA (Mcc)</th>
<th>SST-2 (Acc)</th>
<th>MRPC (Acc)</th>
<th>QNLI (Acc)</th>
<th>MNLI-m/mm (Acc)</th>
<th>QQP (F1)</th>
<th>RTE (Acc)</th>
<th>STS-B (Spc)</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT\textsubscript{base}</td>
<td>51.9</td>
<td>93.5</td>
<td>88.9</td>
<td>90.5</td>
<td>84.6/83.4</td>
<td>71.2</td>
<td>66.4</td>
<td>85.8</td>
<td>79.6</td>
</tr>
<tr>
<td>BERT\textsubscript{base (Ours)}</td>
<td>52.1</td>
<td>92.9</td>
<td>88.7</td>
<td>90.2</td>
<td>84.6/83.4</td>
<td>71.3</td>
<td>67.4</td>
<td>84.6</td>
<td>79.5</td>
</tr>
<tr>
<td>+ Ub</td>
<td>52.0</td>
<td>93.0</td>
<td>89.1</td>
<td>90.6</td>
<td>84.7/83.7</td>
<td>71.5</td>
<td>66.7</td>
<td>85.0</td>
<td>79.7</td>
</tr>
<tr>
<td>+ Gb</td>
<td>52.0</td>
<td>93.6</td>
<td>89.2</td>
<td>90.7</td>
<td>84.5/84.0</td>
<td>71.8</td>
<td>66.9</td>
<td>85.9</td>
<td>79.8</td>
</tr>
<tr>
<td>+ Lb</td>
<td>53.1</td>
<td>93.3</td>
<td>89.7</td>
<td>90.5</td>
<td>84.8/84.4</td>
<td>71.8</td>
<td>67.3</td>
<td>86.0</td>
<td>80.1</td>
</tr>
<tr>
<td>+ MOMETAS</td>
<td>55.9</td>
<td>93.7</td>
<td>90.0</td>
<td>90.7</td>
<td>85.2/84.3</td>
<td>72.1</td>
<td>68.4</td>
<td>86.9</td>
<td>80.8</td>
</tr>
</tbody>
</table>

When pre-training with MOMETAS, the additional time consumption mainly comes from doing evaluation in meta-test. Though it will rise as the number of objectives increases, the evaluation is done only once every $K$ steps (e.g. 100) and is inherently fast with no backward passes. Thus, the overhead brought by MOMETAS is minimal.

4 Experimental Setup

In this section, we present our experimental setup. Our implementations are based on PyTorch using transformers (Wolf et al., 2020).

4.1 Pre-training Objectives

We adopt five pre-training objectives in our experiments. The details of them are listed below.

- **General Language Representation - Masked Language Modeling (MLM):** Following BERT (Devlin et al., 2019), we randomly sample 15% of the tokens in each input sequence and replace them with special [MASK] elements.
- **Semantic - Contrastive Learning of Sentence Embeddings (CSE):** Following SimCSE (Gao et al., 2021), we feed the same sequence twice by applying different dropout masks and extract the [CLS] elements as their sentence representations. The model is required to predict the input sentence itself from in-batch negatives.
- **Coherence - Added Token Detection (ATD):** We randomly sample 15% of the positions in each sequence and insert random tokens in them. The model is required to decide which positions are superfluous. Different from MLM, ATD expands the context of text. Different from MLM, ATD expands the context of text.
- **Entity & Knowledge - Entity-guided Masked Language Modeling (EMLM):** We leverage the prior knowledge to further strengthen the model. We first pick out the entities in each sequence\(^2\) and then randomly replace a half of them with [MASK] (Xiong et al., 2020).
  Though we are unable to cover all alternatives in this paper, the experiments are of great potential to be extended to other pre-training setups.

4.2 Dataset

Based on our pre-training setup, we validate our approach on a wide range of downstream benchmarks (14 tasks in total). In what follows, we summarize them as well as describe how the chosen ones relate to our pre-training objectives.

**General Natural Language Understanding**

We adopt GLUE benchmark (Wang et al., 2019a), a collection of eight natural language understanding tasks, including natural language inference, sentiment analysis and semantic similarity. We exclude problematic WNLI as in Devlin et al. (2019)). In addition, we adopt SICK (Marelli et al., 2014), another natural language inference benchmark as a complement.

**Semantic Similarity** We further adopt PAWS-QQP (Zhang et al., 2019), which adds adversarial examples to QQP for evaluating model robustness. Following the zero-shot setting in Zhang et al. (2019), we train the model on QQP and directly evaluate it on PAWS-QQP.

**Named Entity Recognition (NER)** We adopt two benchmarks, CoNLL-2003 (Sang and Meulder, 2003) and WNUT-2017 (Derczynski et al., 2017). Of these, WNUT-2017 contains a large

\(^2\)https://github.com/stanfordnlp/stanza
Multi-choice Machine Reading Comprehension (MRC) Two challenging benchmarks are adopted. DREAM (Sun et al., 2019) for multi-turn dialogue understanding, and aNLI (Bhagavatula et al., 2020) for commonsense reasoning, both of which are in format of multi-choice MRC.

Notably for DREAM and aNLI, there are no straightforward objectives adopted. However, it is desirable that the model is able to learn the interdisciplinary knowledge and generalize better on tasks not seen during pre-training through jointly learning multiple objectives.

### 4.3 Baseline Strategies

We compare MOMETAS with several earlier discussed sampling strategies, including Uniform-based (Ub), Gradient-based (Gb), and Loss-based (Lb). Experiments are made on BERT\textsubscript{base} models.

Except for Ub, the rest two are based on proportion, that is we sample the objectives as proportional to the magnitudes of concerned values. To implement, we compute the average gradient (L2 norm of gradients over encoder parameters) or loss of each objective for every certain number of training steps (to keep in pace with MOMETAS, also $K$ steps). At the same point as meta-test, we update the distribution. However, we find some large values (e.g. big gradient at the start of training) will make the probabilities of other objectives close to zero. Following Andrychowicz et al. (2016), we use Sigmoid function to scale them properly.

### 4.4 Training Details

**Pre-training** Inherited from the released checkpoints, bert-base-uncased and bert-large-uncased\textsuperscript{3}, we continue to pre-train our models following multi-objective setting. For training corpus, we use a subset of Colossal Clean Crawled Corpus (Raffel et al., 2020) (we use nearly 100GB of it and randomly sample 1GB for validation). Each single model is trained with 512 batch size and for 50K steps (nearly one epoch). Unless otherwise specified, we fix meta length $K$ to 100 and meta step size to 1e-1. Training a base/large-size model takes about 12/36 hours on 8 V100 GPUs with FP16 for both uniform-based sampling and MOMETAS.

**Fine-tuning** For all GLUE sub-tasks, we follow the hyperparameters shared in Lan et al. (2020) and fine-tune for 3 epochs, except 10 epochs for

---

\textsuperscript{3}https://github.com/huggingface/transformers/
RTE and STS-B. For other tasks, we merely sweep through learning rates and batch sizes for efficiency, excluding dropout probabilities or weight decay rates. Readers can refer to Appendix A for details.

5 Empirical Results

GLUE Table 1 reports the test results on GLUE benchmark under different sampling strategies, all of which are based BERT base. Intuitively, simple uniform multi-objective pre-training (Ub) merely leads to limited performance gain (79.5 → 79.7). Besides, we find that Gb is also not effective, while Lb brings nice gain (79.5 → 80.1). However, more powerful performance gain can be seen on MOMETAS-empowered one (79.5 → 80.8). Compared to Ub, MOMETAS outperforms it on all eight sub-tasks (3.9 points absolute gain on CoLA, 0.7 on SST-2, 0.9 on MRPC, 1.7 on RTE, 2.3 on STS-B), which indicates the strength of our meta-learning-based sampling.

More tasks We make further experiments on more different tasks as in Table 2. Generally, MOMETAS better facilitates multi-objective pre-training compared to Ub and Lb. We first focus on two semantic similarity tasks (STS-B and PAWS-QQP), for which we adopt CSE to improve the performance. According to Gao et al. (2021), single CSE-trained BERT can achieve significant improvement. When the number of objectives increases, however, the situation can be difficult. It does not work well with Ub (84.8 → 85.2 on STS-B). Contrarily, MOMETAS brings a huge performance boost on BERT base (84.8 → 86.5 on STS-B, 33.4 → 36.5 on P-QQP), even surpasses BERT large. Similar situation can be found on NER comparing Ub with MOMETAS (50.8 → 52.1 on WNUT). It demonstrates that MOMETAS helps maintain the benefit of a single objective in the multi-objective scenario. Additionally, MOMETAS-empowered BERT base is able to outperform BERT large on some of the tasks (SICK, P-QQP, STS-B, CoNLL and Wanut), suggesting the great potential of multi-objective pre-training. On the other hand, because of the attempt to learning cross knowledge from other objectives, MOMETAS also enables the model to learn well on MRC tasks, even though there are no related objectives adopted.

Single-objective Table 2 also demonstrates the superiority of multi-objective pre-training over single-objective. We find that the overall performance gain brought by each single objective is limited. Though it may lead to notable improvement on certain tasks (e.g. ELM on NER), it may also cause the model to perform particularly badly on others (CSE on NER). However, MOMETAS is designed to find the all-round direction where the model is able to perform well on all objectives.

6 Visualization

Probability distribution Figure 2 depicts the sampling weights averaged throughout the training process of all pre-training objectives learned by MOMETAS. Intuitively, the distribution looks more volatile when \( \lambda = 2 \) (upper), while more clustered when \( \lambda = 3 \) (bottom), which indicates the role of entropy regularization. From both cases, we may find some common clues. DTD always stands a high picking weight, which uncovers the potential of deleting corruption when learning a denoising encoder. In addition, both MLM and ELM are never underweight, and a general masking strategy outweighs a specific one. Then we look at CSE, a sentence-level objective, which is much easier than the other token-level ones. We find its picking weight is very high in the early period of training but drops quickly in the later period.

Reward We observe the respective reward curves of MOMETAS and Ub to access to their training gain for multi-objective pre-training. To make intuitive, we depict the difference of them (the former minus the latter) as in Figure 3. Intuitively, we see slight differences at the beginning of training since MOMETAS is initialized with uni-
form distribution. However, all three curves are positive for majority of the time. When $\lambda = 1$ for instance, we see a rising trend of the curve, from negative to positive, while when $\lambda = 3$, the curve is always above zero, which implies that MOMETAS learns to achieve more evaluation scores than Ub in meta-test.

## 7 Ablation Studies

This section reports our ablation studies over a number of factors of MOMETAS in order to better understand their roles. For all experiments, we report the results over five runs.

### 7.1 Comparison between Rewarding Functions

We compare different rewarding functions $R(\tau)$ on three GLUE sub-tasks, SST-2, QNLI and STS-B: (1) **overall loss rewarding**: we optimize the summation of all losses; (2) **relative individual rewarding**: exactly what we use in MOMETAS, we optimize the summation of all relative loss drops as Eq. 3; (3) **hard individual rewarding**: similar as the relative one, we replace the individual loss drop with $\pm 1$ when it is down or up respectively and optimize the summation of them.

As shown in Table 3, slight improvement can be seen when simply rewarding MOMETAS with overall loss compared to uniform-based sampling in Table 1. In this situation, it is hard to learn the balance between all objectives. However, individual rewarding can achieve stronger performances in both hard and relative cases.

### 7.2 Effect of Entropy Regularization

When optimizing MOMETAS, we apply maximum entropy regularization to encourage exploration in the hope of seeking out the global optima. Table 4 demonstrates the effect of different degrees of entropy regularization on pre-training performances. We can see general gain compared to original BERT in Table 1 even if there is no regularization applied. However, regularization further boosts the performances. The best case occurs when $\lambda = 3$, which the model outperforms the base one by 0.5, 0.7 and 1.1 points on all three tasks, respectively.

### 7.3 Effect of Meta Length

In our pre-training framework, MOMETAS is designed to be updated every $K$ steps. $K$ refers to the number of steps of meta-train and meanwhile reflects the knowledge accumulation before meta-test. Generally, when $K$ becomes larger, MOMETAS tends to be less sensitive and pay more attention to long-term benefits. Contrarily, when $K$ is close to 1, it is greedy and only cares about the current moment. In practical, it cannot be smaller than the number of objectives.

Table 5 shows the pre-training performances under a number of values of $K$. We can see that a too small $K$ may lead to worse results (e.g. $K = 25$). It can be presumed that **long-sight helps to find the global optimum**. For example, we cannot acquire sufficient meta knowledge to justify all objectives when $K$ is too small. This can be supported by another fact that **MOMETAS is found more uniform-distributed when $K$ becomes smaller** under the same degree of entropy regularization. On the other hand, we can see nice results when

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>MNLI-m</th>
<th>STS-B</th>
<th>WNUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>84.7(0)</td>
<td>85.8(3)</td>
<td>51.0(6)</td>
</tr>
<tr>
<td>1</td>
<td>85.1(1)</td>
<td>86.2(2)</td>
<td>51.7(6)</td>
</tr>
<tr>
<td>2</td>
<td>85.3(2)</td>
<td>86.2(5)</td>
<td>50.8(2)</td>
</tr>
<tr>
<td>3</td>
<td>85.2(2)</td>
<td>86.5(2)</td>
<td>52.1(7)</td>
</tr>
</tbody>
</table>

Table 4: Effect of entropy regularization on BERT$_{base}$. The base model is trained with no regularization.
Table 5: Effect of meta length on BERT$_{base}$. Note that the results are based on five runs but we do not list the variances for space limitation.

<table>
<thead>
<tr>
<th>K</th>
<th>MNLI-m</th>
<th>SICK</th>
<th>WNUT</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>84.6</td>
<td>87.5</td>
<td>51.3</td>
<td>74.5</td>
</tr>
<tr>
<td>50</td>
<td>85.1</td>
<td>87.5</td>
<td>51.7</td>
<td>74.8</td>
</tr>
<tr>
<td>100</td>
<td><strong>85.2</strong></td>
<td><strong>87.9</strong></td>
<td>52.1</td>
<td><strong>75.1</strong></td>
</tr>
<tr>
<td>200</td>
<td>85.0</td>
<td>87.7</td>
<td><strong>52.4</strong></td>
<td>75.0</td>
</tr>
</tbody>
</table>

8 Conclusion

This paper concentrates on multi-objective pre-training of PrLMs and presents Multi-Objective Meta-Sampler (MOMETAS) in the hope of combining arbitrary pre-training objectives organically. We adopt five pre-training objectives and conduct experiments on the base-size and large-size models. The empirical results on a wide range of NLP tasks demonstrate that MOMETAS largely outperforms other rule-based sampling strategies and unlocks more powerful language models.

Limitations

This paper proposes a novel pre-training framework, and therefore requires larger GPU resources. However, we will release our trained checkpoints, pre-training corpus, and code to facilitate further research. Our pre-training experiments are limited in continual pre-training as there are only 8 GPUs available. We therefore expect future researchers to practice and validate our approach when pre-training from scratch, even on stronger model architectures.

This paper discusses fewer on how we choose each single pre-training objective. We do not rule out other potential options that can make the pre-trained model even better. Of course, we will keep following up on this part of the study and train new models. Besides, we do not make the experiments with a larger number of objectives (e.g. 10). It is possible that the optimized value of $\lambda$ for entropy regularization will be different when the number becomes larger.

Another limitation is that we do not discuss the role of the validation set which is necessary for meta-learning. Intuitively, a carefully-selected validation set may improve the credibility of meta-test. For example, it can be positive to introduce signals that are more related to the downstream tasks. We will leave this part for our future work.

References


Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. (DEBERTA): [DECODING]-[enhanced] [bert] [with] [disentangled] [attention]. In International Conference on Learning Representations.


A Training Details
<table>
<thead>
<tr>
<th>Number of hidden layers</th>
<th>BERT\textsubscript{base}</th>
<th>BERT\textsubscript{large}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hidden size</td>
<td>12</td>
<td>24</td>
</tr>
<tr>
<td>Intermediate size</td>
<td>768</td>
<td>1024</td>
</tr>
<tr>
<td>Number of attention heads</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Batch size</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>Learning rate</td>
<td>5e-5</td>
<td>5e-5</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>Max sequence length</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>Warmup proportion</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>Max steps</td>
<td>50K</td>
<td>50K</td>
</tr>
<tr>
<td>Gradient clipping</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>FP16</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Number of GPUs</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Training period</td>
<td>12 hours</td>
<td>36 hours</td>
</tr>
</tbody>
</table>

Table 6: Hyperparameters for pre-training.

<table>
<thead>
<tr>
<th>MNLI</th>
<th>SICK</th>
<th>QQP</th>
<th>STS-B</th>
<th>CoNLL</th>
<th>WNUT</th>
<th>DREAM</th>
<th>aNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dropout</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Batch size</td>
<td>128</td>
<td>32</td>
<td>128</td>
<td>16</td>
<td>32</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Learning rate</td>
<td>3e-5</td>
<td>5e-5</td>
<td>5e-5</td>
<td>5e-5</td>
<td>5e-5</td>
<td>5e-5</td>
<td>3e-5</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>Max sequence length</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>64</td>
<td>128</td>
</tr>
<tr>
<td>Warmup proportion</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.06</td>
<td>0.1</td>
<td>0.1</td>
<td>0.06</td>
</tr>
<tr>
<td>Max epochs</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>3</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>FP16</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 7: Hyperparameters for fine-tuning.