Adaptør: Objective-Centric Adaptation Framework for Language Models

Michal Štefánik[1,2] and **Vít Novotný**[1] and **Nikola Groverová**[2] and **Petr Sojka**[1]
[1]Faculty of Informatics, Masaryk University, Czech Republic
[2]Gauss Algorithmic

Abstract

Progress in natural language processing research is catalyzed by the possibilities given by the widespread software frameworks. This paper introduces the Adaptør library[1] that transposes the traditional model-centric approach composed of pre-training + fine-tuning steps to objective-centric approach, composing the training process by *applications* of selected *objectives*. We survey research directions that can benefit from enhanced objective-centric experimentation in multi-task training, custom objectives development, dynamic training curricula, or domain adaptation. Adaptør aims to ease the reproducibility of these research directions in practice. Finally, we demonstrate the practical applicability of Adaptør in selected unsupervised domain adaptation scenarios.

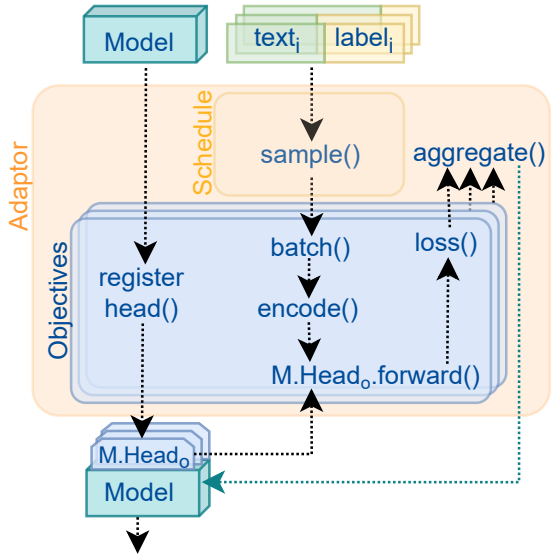

Figure 1: Overview of Adaptør's objective-centric training framework: Objective 1) registers its compatible head on top of the shared model, 2) performs specific input encoding, and 3) compute loss value based on its output. A Schedule implements a specific sampling curricula and Adaptør aggregates and propagates objectives' losses and performs optimization.

"The measure of intelligence is the ability to change."
— Albert Einstein

1 Introduction

Recent development in Natural Language Processing (NLP) heavily benefits from a high level of maturity of open-source frameworks, such as Fairseq (Ott et al., 2019) or HuggingFace Transformers (Wolf et al., 2020). Thanks to the standardized interfaces, these libraries allow for immediate experimentation with the most recent research results, practically fostering the speed of further progress in the area. While their use is seamless for countless conventional use-cases of transformer models and fine-tuning to a specific end-task (Devlin et al., 2019; Radford and Narasimhan, 2018), divergence from this framework requires feasible, but elaborate and complex customizations, increasing the risk of logical errors and complicating the reproducibility of experiments. A characteristic group of problems requiring significant changes to the standard pipeline are multi-step and multi-task adaptations.

This paper introduces the Adaptør library, which aims to simplify the more complex training processes that their training objectives can easier describe. Adaptør challenges the conventional *model-centric* framework, where data and task selection are constrained by the requirements of the selected language model architecture. Instead, it introduces an *objective-centric* training pipeline, with Objective as the central abstraction of the process.

The Adaptør framework aims to help NLP researchers and practicioners engage in projects that include any of the following:

- **Multi-objective training**: when training a language model on more than one task or data set, including languages, Adaptør can signif-

[1] github.com/gaussalgo/adaptor

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
System Demonstrations, pages 261 - 269

icantly simplify the custom code base that needs to be implemented. Even if the objective is custom, the user can avoid adjustments to other parts of the training pipeline.

- **Custom data schedule**: when users need to perform dynamic data sampling, Adaptør allows them to implement a custom Schedule (see Figure 2), leaving the data and model adjustment logic intact. This simplifies systematic experimentation and reproducibility, and minimizes the risk of errors.

- **Objectives design & evaluation**: Adaptør exposes top-level declaration of training objectives, which enables easy experimentation with custom objectives. Objective-level monitoring can provide custom behavioural insights and allows for pruning less promising experiments earlier in the lengthy training process, saving computational costs.

- **Robustness evaluation**: The objective-centric paradigm provides an easy robustness estimation by evaluating on out-of-distribution samples. In the standard *sequential* adaptation scenario, objective-centric evaluation exposes characteristic flaws of adaptation, like exposure bias or catastrophic forgetting.

This paper is structured as follows: Section 2 provides an overview of recent work demonstrating the potential of multi-objective training in domain and task adaptation. Section 2.4 also describes other software frameworks applicable for similar use cases. Section 3 describes the design of Adaptør, showing the users how to confidently integrate novel objectives and schedules. In Section 4, we describe and implement a set of non-trivial, yet promising domain adaptation experiments using Adaptør and collect their results. As Adaptør remains under active development, we close in Section 5 with an outline of the upcoming features. We welcome contributions of novel objectives and schedules.

2 Background

This section provides an overview of recent work that demonstrates the potential of multi-objective training and schedules that motivated the design of Adaptør. Our overview consists of a non-exhaustive list of applications that Adaptør aims to make more accessible for practical use and in future research.

2.1 Multi-Task Training

Multi-task training has a long history in both traditional machine learning (Caruana, 1997) and in deep learning (Crawshaw, 2020). This section describes examples of multi-task (i.e. multi-objective) training, outlining its benefits and potential.

Under some circumstances, multi-task training enhances distributional robustness of neural models. Tu et al. (2020) demonstrate this on adversarial data sets, exposing common heuristic biases of the language models (McCoy et al., 2019). Enhanced model generalization can also be achieved by introducing one or more latent tasks that do not directly correspond to the end task but reflect specific desired properties of the model. One of a few studies in this direction is Sharpness-Aware Minimisation of Foret et al. (2021), performing multi-objective training on image classification using cross-entropy and a novel, sharpness-aware objective, reflecting the model's monotonicity on the local neighborhood. In context of Neural Machine Translation (NMT), Wang and Sennrich (2020) incorporate Minimum Risk Training (MRT) objective (Ranzato et al., 2016), optimising an arbitrary sequence-level measure of outputs. In composition with the traditional token-level cross-entropy objective, MRT improves distributional robustness.

By aggregating multiple objectives, Xie et al. (2019) show that combining sentence classification objective with maximizing representation consistency to augmented samples fosters data efficiency.

The intuition on the benefits of multi-task training presumes that by optimizing the training by multiple cost functions, the final model is less prone to the weaknesses of a specific task (Collobert et al., 2011), possibly reflecting on higher-level, task-invariant properties of language (Bengio et al., 2013).

2.2 Data-Sampling Schedules

Exposing a model to training samples in a systematic schedule, also referred to as a *curriculum*, can lead to an improvement of the accuracy of the final model (Bengio et al., 2009). While the positive effects of more complex schedules based on sample "difficulty" with transformers remain to be explored, multiple studies show the potential of confidence-based sampling to improve accuracy

and generalization. Biased samples can be identified, according to model's confidence (Pleiss et al., 2020; Swayamdipta et al., 2020) or using Bayesian methods such as the Product of Experts (Hinton, 2002). Then, they can be either eliminated (Bras et al., 2020) or downweighted (Utama et al., 2020).

More complex scheduling methods are applied in training NMT models. Bengio et al. (2015) use decay schedule to sample from both references and the previous outputs of a NMT model, minimizing the discrepancy between training and inference. Zhang et al. (2019) successfully use the same sampling strategy in a sequence-level objective. The results of Lu et al. (2020) underline the potential of sampling in NMT training, suggesting that the accuracy of transformers on reported MT benchmarks can be outperformed by simpler RNN models by combining objectives in decay schedule.

Despite the reported improvements, we find that custom scheduling strategies are rarely used. We attribute this to their complicated integration into the standard training process. To foster the research and applicability of scheduling methods, Adaptor makes the implementation of custom scheduling strategies easy, comprehensible, and reproducible.

2.3 Domain Adaptation

Objective-centric frameworks are well-suited for domain adaptation techniques, where Adaptor provides support for combining traditional end-task objectives with unsupervised adaptation or auxiliary-task objectives in a user-selected schedule. The goal of domain adaptation is to maximize performance on a specific data domain, often denoted as the *adapted* or *target domain* (Saunders, 2021).

Perhaps the most common adaptation approach using pre-trained language models is to continue pre-training on unsupervised samples of the adapted domain (Luong and Manning, 2015; Lee et al., 2019; Beltagy et al., 2019). This approach has been successfully extended in various directions. For instance, Gururangan et al. (2020) show that adapting to a shared task on different domain can enhance accuracy of the eventual application. If supervised data is sparse, other auxiliary tasks, described earlier in Section 2.1, can be used as concurrent objectives (Xie et al., 2019).

In cases where larger volumes of data of given task is available in a different language, adaptation using cross-lingual transfer can be considered. Pre-trained language models show that cross-lingual transfer works well with large-data unsupervised objectives (Conneau and Lample, 2019), but it can also be applied for low-resource supervised objective, such as very low-resource translation (Neubig and Hu, 2018).

If even unsupervised target-domain data is sparse, another option is to subset arbitrary unsupervised sources to automatically identify samples of adapted domain, by applying domain classifier (Jiang and Zhai, 2007; Elsahar and Gallé, 2019). If the boundary between the training and the adapted domain is known, an auxiliary objective can minimise a discrepancy of representations between the training and possibly low-resource target domain (Chadha and Andreopoulos, 2018).

Despite the possibilities, adaptation can also introduce undesired biases. In the scope of NMT, adaptation can cause problems of "catastrophic forgetting", when the model experiences performance degradation on the originally well-performing domains (Saunders, 2021), or "exposure bias", when the model overfits the non-representative specifics of the target domain, such as the artifacts of data collection (Ranzato et al., 2016). Additionally, by normalizing a single type of bias, such as lexical overlap (McCoy et al., 2019), the model might degrade its accuracy on other domains (Utama et al., 2020). Addressing multiple biases concurrently (Wu et al., 2020) can mitigate this problem.

Adaptor allows the knowledgeable user to construct a reproducible and robust adaptation pipeline using native multi-objective evaluation. Covering multiple domains in separate objectives, Adaptor can expose the above pitfalls, without the need to implement complex separate evaluation routines.

2.4 Related Software Frameworks

The *Adapters* architecture (Houlsby et al., 2019), having only a small set of parameters, might be a good fit when performing adaptation of transformer with modest hardware or data. Recently, the AdapterHub library (Pfeiffer et al., 2020) makes training and sharing of Adapters convenient. Compared to Adaptor, AdapterHub does not provide support for more complex adaptation cases, such as using multiple objectives, scheduling, or extended evaluation. However, since both libraries build upon the HuggingFace Transformers library (Wolf et al., 2020), their close integration is feasible.

If the robustness of models to heuristic shortcuts (McCoy et al., 2019) is the primary goal, the

```python
class ParallelSchedule(Schedule):
    def _sample_objectives(self, split: str) -> Iterator[Objective]:
        while True:
            for objective in self.objectives[split].values():
                yield objective
```

Figure 2: Adaptør provides a convenient base for implementing custom sampling schedules. ParallelSchedule in the figure demonstrates an implementation of the schedule sampling the update objectives in rotation. Further, the sampling can be easily conditioned on the state of Objectives such as the recent outputs, loss, or metrics evaluations.

Robustness Gym library (Goel et al., 2021) provides a comprehensive evaluation over an extendable set of different kinds of heuristic biases. Robustness Gym provides much deeper evaluation compared to Adaptør Evaluators, and could be integrated as an Adaptør Evaluator. Unlike Robustness Gym, Adaptør enables an evaluation of robustness also on generative tasks, with specified out-of-domain data sets.

3 Adaptør Design

This section describes the structure and functions of the Adaptør framework. We introduce its primary components bottom-up. Figure 3 depicts the relations of these components and compares user interaction with the traditional model-centric pipeline.

3.1 LangModule

A LangModule instance provides a management of inputs, outputs and objective-specific model components, referred to as *heads*. Once an objective with given LangModule is instantiated, an objective-compatible model is either initialised, or given by the user (see Section 3.2) and the parameters of this model are merged with the parameters of the previously-registered objectives.

The merge works as follows: If no previous objective was registered, then the model of the given objective is considered a base model. The models of the second- and later-registered objectives are then merged with the base model: first, pairs of PyTorch modules of the same name in the base and the new model are identified. If the dimensions and weights of these modules match, the respective module of the newly-adding model is replaced with a module of the base model.

In the case of pre-trained transformers, the weights of heads are initialized randomly by default, resulting in a registration of a distinct head for each objective and sharing the remaining parameters. Users can control which parameters (not) to merge by explicitly setting their respective weights as (non-)equal.

It is possible to use LangModule with any PyTorch module that uses a HuggingFace tokenizer, compatible with the given neural module. Therefore, LangModule is also suitable for other models such as recurrent networks.

3.2 Objective

Objectives are the primary component of Adaptør's training pipeline. Most importantly, an Objective serves two functions: sample encoding and loss computation. By implementing these and choosing the type of a model's head, Adaptør users can define and experiment with novel training objectives. If they additionally provide an explicit definition of the Objective's model (the `objective_module` attribute), the new objective does not even have to comply with common model heads; shared parameters of the given `objective_module` would still be merged with the given `lang_module`.

If no `objective_module` is given, the Objective will request that a LangModule assigns the Objective a module of the Objective's default `compatible_head` (see Section 3.1).

Additionally, every Objective instance performs its own logging, evaluation, and state updates, such as its convergence, based on a valuation of given `val_evaluators`, or draws a progress bar, based on the state of its sample iteration. However, the training flow is guided by a Schedule (see Section 3.3). Objectives can implement custom data sampling, but if possible, we recommended to do so in a custom Schedule instance.

Since data encoding is also objective-specific, Objectives expose a higher-level user interface of data inputs than other frameworks: instead of encodings, users provide an Objective with a `texts_or_path` and a `labels_or_path` containing raw texts and respective labels. Adaptør provides an implementation of standard Objectives for sequence and token classification and sequence-to-sequence tasks. When implementing a custom Objective, note that sampling and encoding are performance bottlenecks on current high-end GPUs.

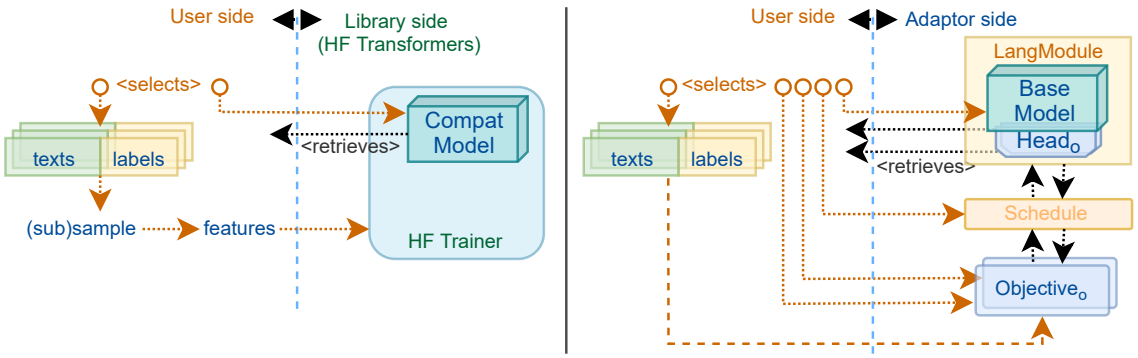

Figure 3: A comparison of interaction with a model-centric HuggingFace Trainer (left) and objective-centric Adaptor (right): While in model-centric approach, user resolves text processing, sampling and encoding compatible with selected model of specific objective, objective-centric approach delegates these functionalities to Objective instances. Explicit definition of Objectives and Schedule on Adaptor's user side makes otherwise complex multi-objective and custom-schedule experiments transparent and reproducible.

3.3 Schedule

Schedules control the training flow through the interfaces provided by HuggingFace Transformers library. Primarily, they deliver 1) a set of standard stopping strategies based on the state of the Objectives and 2) an IterableDataset instance, constructed by sampling Objectives according to a sampling strategy implemented in its `_sample_objectives`. A Schedule also ensures that outputs of distinct `lang_modules`' heads are delivered to the respective Objectives for loss computation.

This relatively complex sampling framework provides a very simple interface for custom Schedule implementations (see Section 2.2). For instance, a pre-defined `ParallelSchedule` is implemented with three lines of code (see Figure 2).

3.4 Adapter

An Adapter is customization of the HuggingFace Trainer with only minor adjustments. Specifically, Adapter redirects loss computation to a Schedule, which further distributes outputs to corresponding Objectives and extends native training logs with logs of Objectives' Evaluators. Furthermore, Adapter adjusts persistence of the models so that a model of every head can be reloaded without the use of Adaptor, by simply using HuggingFace Transformers' `AutoModelForXY.from_pretrained`.

Based on the actively-developed HuggingFace Transformers library, the Adaptor allows its users to benefit from all other native features of HuggingFace Transformers, such as the support for the most recent models, custom logging platforms, or distributed parallel training. Furthermore, it can simplify integration with other custom libraries (see Section 2.4).

4 Experiments

We use Adaptor in a set of domain adaptation experiments for a machine translation use-case, aiming to answer the following research question: **How well can unsupervised objective(s) substitute labeled parallel data**. In our methodology, we permute the easily-configurable parts of Adaptor's training configuration[2] and compare the results of the resulting model to a baseline adaptation scenario. We experiment with an architecture identical to the base model of Vaswani et al. (2017), with a configuration of Junczys-Dowmunt et al. (2018).

Data. We train the model on English-to-Czech translations on different domains of OPUS (Tiedemann, 2012) chosen for their significant distinctiveness: we use *Wikimedia* as a large-scale, supervised domain (denoted as *in-domain*, i.e. ID), *OpenSubtitles* as an Adapted Domain (AD) and *Bible* for the evaluation of a model's robustness on Out-Of-Domain (OOD) samples.

Pre-training vs. fine-tuning. We simulate two basic scenarios: training the model from a random initialization and fine-tuning the existing translation model with no control over its pre-training data. In the latter cases, we perform fine-tuning from the checkpoint of Tiedemann and Thottingal (2020).

Schedules. We implement and experiment with two objective schedules: i) **Sequential** schedule, sampling and differentiating the model sequentially

[2]Our code is available on https://github.com/gaussalgo/adaptor/tree/reprod/demo.py

	Schedule	Objectives	BLEU_{ID}	BLEU_{AD}	BLEU_{OOD}	BERTS_{ID}	BERTS_{AD}	$\text{BERTS}_{\text{OOD}}$
Pre-training		***1)*** $\text{Seq2Seq}_{\text{ID}}$	28.18	5.34	0.91	0.833	0.738	0.671
	Sequent.	***2)*** $\text{Seq2Seq}_{\text{ID}}$+ $\text{BackTr}_{\text{AD}}$	5.10	15.01	2.57	0.740	0.805	0.733
		****3)*** $\text{Seq2Seq}_{\text{ID}}$+ $\text{Seq2Seq}_{\text{AD}}$	4.96	17.37	2.64	0.756	0.816	0.726
	Parallel	***4)*** $\text{Seq2Seq}_{\text{ID}}$+ $\text{BackTr}_{\text{AD}}$	31.06	16.99	2.46	0.852	0.817	0.722
		****5)*** $\text{Seq2Seq}_{\text{ID}}$+ $\text{Seq2Seq}_{\text{AD}}$	29.72	18.55	2.98	0.843	0.813	0.732
Fine-tuning		***6)*** $\text{Seq2Seq}_{\text{ID}}$	37.97	17.62	6.50	0.875	0.808	0.758
		7) $\text{BackTr}_{\text{AD}}$	30.34	22.98	11.08	0.869	0.834	0.799
	Parallel	***8)*** $\text{Seq2Seq}_{\text{ID}}$+ $\text{Denois}_{\text{AD}}$	38.96	13.37	6.87	0.876	0.782	0.761
		9) $\text{Seq2Seq}_{\text{ID}}$+ $\text{BackTr}_{\text{AD}}$	38.25	21.47	9.03	0.873	0.831	0.791
		****10)*** $\text{Seq2Seq}_{\text{ID}}$+ $\text{Seq2Seq}_{\text{AD}}$	40.72	23.35	6.97	0.880	0.836	0.772

Table 1: We evaluate the features of Adaptør on multi-objective domain adaptation in machine translation: our experiments compare the BLEU score and BERTScore of unsupervised adaptation (*Seq2seq* + *Denoising* or *Back-Translation*) applied in different schedules, to *no* adaptation ***(1, 6)*** and a hypothetical supervised adaptation ***(*3, *5, *10)***. Results show that the Parallel schedule eliminates *catastrophic forgetting* and that unsupervised Back-translation is able to reach performance that is close to the supervised adaptation.

by each objective until convergence by evaluation loss, or for a maximum of 100,000 updates. ii) **Parallel** schedule, concurrently sampling training batches uniformly from every given objective. Using gradient accumulation, we differentiate the model based on *all* given objectives. We perform updates until the convergence of *all* objectives, or for a maximum of 50,000 updates for each objective.

Objectives selection. We implement and experiment with the following Adaptør objectives:

- **Sequence-to-sequence** (seq2seq) objective, as introduced by Vaswani et al. (2017), maps a combination of encoder inputs in the source language and previously-generated outputs as decoder inputs to a distribution over the next-predicted tokens.

- **Denoising** objective introduced by Lewis et al. (2020) is an unsupervised instance of the seq2seq objective that performs random token permutation on the input and trains the model to map such 'noisy' text to the original version of the input. We use this objective on the target-data domain to enhance its comprehension by the model.

- **Back-translation** objective, as used e.g. by Sennrich et al. (2016) is also an unsupervised seq2seq objective, which uses an external translator in reverse direction to obtain pseudo-inputs. This objective is profitable when we have unlabeled data of the target domain.

Using these components, we construct the following experiments:

- **Baselines:** pre-training ***(1)*** and fine-tuning ***(6)*** on ID data from a domain different from the Application Domain (AD) using a single traditional *seq2seq* objective.

- **Sequential adaptation:** we pre-train using *seq2seq* on ID and afterwards adapt using either unsupervised *Back-translation* ***(2)***, or supervised *seq2seq* ***(3)*** on AD to quantify the unsupervised adaptation gap.

- **Parallel adaptation:** we concurrently train on both *seq2seq* and another unsupervised objective: *Back-translation* ***(4, 9)*** and *Denoising* ***(8)***. Again, we compare the gap to the supervised situation ***(5, 10)***.

4.1 Results

Table 1 evaluates the base transformer after the given number of updates on held-out deduplicated validation splits of In-Domain (ID), Adapted-Domain (AD), and the third Out-Of-Domain (OOD) data. Note that the results for the BLEU score are properly comparable only within the same domain.

We observe that the model trained on a single domain ***(1, 6)*** degrades on *all* other domains. In a pre-training scenario, domain robustness improves when incorporating data of adapted domain in *any* objective. However, in a sequential schedule, we observe catastrophic forgetting towards any most-recent domain of adaptation ***(2, 3)***. This is improved by using the Parallel schedule for a negligible price of in-domain accuracy ***(4, 5)***.

In the fine-tuning scenario, we show that incorporating unsupervised Back-translation to AD ***(7, 9)*** improves ID BLEU comparably to supervised adaptation ***(10)***. Interestingly, Denoising on AD ***(8)*** improves in-domain performance but seems less efficient than Back-translation.

4.2 Adaptør Usage Complexity

To give an idea about the relative complexity of using Adaptør as compared to model-centric frameworks, we compare selected measurable code features of the complexity of our experimental implementation to an example implementation using the HuggingFace Trainer[3]. We pick the experiment of supervised pre-training + unsupervised fine-tuning, including evaluation, in the sequential schedule ***(2)***, as this can still be addressed using HuggingFace Transformers relatively easily; Implementing the parallel multi-objective schedule in the Transformers framework would require major customisations of selected model and Trainer objects.

The training script using HuggingFace Trainer contains 654 lines of code, 135 variable assignments, 186 method calls and the initialisation of 9 custom objects. Additionally, in the pre-training + fine-tuning framework, this script has to be run twice, initialising the second training from the selected checkpoint of the first one, with updated configurations. Back-translated pseudo-labels are generated by a different script, not included in this assessment.

Using Adaptør, we construct an equivalent routine from the provided demo script. Our implementation contains 124 lines of code, 31 variable assignments, 37 method calls and the initialisation of 14 custom objects. Despite its brevity, our script wraps the whole training process, and hence, together with the associated version of Adaptør or its fork, it provides a reproducible fingerprint of the experiment.

5 Conclusion and Future Work

This paper introduces the Adaptør library, which provides objective-centric training framework well-suitable for multi-task and multi-domain training scenarios, and the development of novel objectives and sampling schedules. We find that even in the conventional single-objective training routines, Adaptør can reduce volumes of custom implementation and increases readability and reproducibility. Having used Adaptør already for several production use cases, we are happy to share it with the NLP community.

Our future work aims to further enhance Adaptør's user comfort with existing and novel unsupervised objectives, dynamic schedules, and demonstrations on novel use cases.

6 Broader Impact

Thanks to the ubiquity of objective-centric training, Adaptør can accelerate the applicability of the most recent research in multi-task and multilingual modeling and enrich the research with the practical experience of the industry.

We further identify the benefits of Adaptør's definite training pipelines in saving unnecessary financial and environmental expenses of reproducing the reported results of large language models, otherwise often including expensive hyperparameter optimization over unreported parameters. Due to these aspects, Adaptør could also ease the spread of state-of-the-art language technologies to under-resourced languages and more specialized domains with a sufficient amount of unsupervised sources.

Finally, objective-centric training might help expose the potential of unsupervised objectives to the generalization and interpretability of models. Adaptør can foster the research in unsupervised learning by lowering the relatively high entry level of technical proficiency needed for experimentation with novel language objectives.

[3]For reference, we use *run_translation.py* example script on HuggingFace Transformers GitHub, version 4.17.0.

References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *Proc. of the Conference on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. ACL.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:1798–1828.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In

Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, pages 41–48, New York, NY, USA. ACM.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew E. Peters, Ashish Sabharwal, and Yejin Choi. 2020. Adversarial filters of dataset biases. In *ICML*.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28:41–75.

A. Chadha and Y. Andreopoulos. 2018. Improving Adversarial Discriminative Domain Adaptation. *CoRR*, abs/1809.03625v3.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.*, 999888:2493–2537.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual Language Model Pretraining. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS)*, Red Hook, NY, USA. Curran Associates Inc.

Michael Crawshaw. 2020. Multi-Task Learning with Deep Neural Networks: A Survey. *CoRR*, abs/2009.09796.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of the 2019 Conference of the North American Chapter of the ACL: Human Language Technologies*, pages 4171–4186, Minneapolis, USA. ACL.

Hady Elsahar and Matthias Gallé. 2019. To Annotate or Not? Predicting Performance Drop under Domain Shift. In *Proc. of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2163–2173, Hong Kong, China. ACL.

Pierre Foret, Ariel Kleiner, H. Mobahi, and Behnam Neyshabur. 2021. Sharpness-Aware Minimization for Efficiently Improving Generalization. *CoRR*, abs/2010.01412v1.

Karan Goel, Nazneen Fatema Rajani, Jesse Vig, Zachary Taschdjian, Mohit Bansal, and Christopher Ré. 2021. Robustness gym: Unifying the NLP evaluation landscape. In *Proceedings of the 2021 Conference of the North American Chapter of the ACL: Human Language Technologies: Demonstrations*, pages 42–55, Online. ACL.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *Proc. of the 58th Annual Meeting of the ACL*, pages 8342–8360. ACL.

Geoffrey E. Hinton. 2002. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800.

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 *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Jing Jiang and ChengXiang Zhai. 2007. Instance Weighting for Domain Adaptation in NLP. In *Proc. of the 45th Annual Meeting of the ACL*, pages 264–271, Prague, Czech Republic. ACL.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. ACL.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proc. of the 58th Annual Meeting of the ACL*, pages 7871–7880.

Wenjie Lu, Leiying Zhou, Gongshen Liu, and Quanhai Zhang. 2020. A mixed learning objective for neural machine translation. In *Proceedings of the 19th Chinese National Conference on Computational Linguistics*, pages 974–983, Haikou, China. Chinese Information Processing Society of China.

Minh-Thang Luong and Christopher Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the 12th International Workshop on Spoken Language Translation: Evaluation Campaign*, pages 76–79, Da Nang, Vietnam.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proc. of the 57th Annual Meeting of the ACL*, pages 3428–3448, Florence, Italy. ACL.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880, Brussels, Belgium. ACL.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54. ACL.

Geoff Pleiss, Tianyi Zhang 0007, Ethan R. Elenberg, and Kilian Q. Weinberger. 2020. Identifying mislabeled data using the area under the margin ranking. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Alec Radford and Karthik Narasimhan. 2018. Improving Language Understanding by Generative Pre-Training.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence Level Training with Recurrent Neural Networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings*.

Danielle Saunders. 2021. Domain Adaptation and Multi-Domain Adaptation for Neural Machine Translation: A Survey. *CoRR*, abs/2104.06951.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. ACL.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. ACL.

Jörg Tiedemann. 2012. Parallel Data, Tools and Interfaces in OPUS. In *Proc. of the Eighth International Conf. LREC*, pages 2214–2218, Istanbul, Turkey. ELRA.

Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT – building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, Lisboa, Portugal. European Association for Machine Translation.

Lifu Tu, Garima Lalwani, Spandana Gella, and He He. 2020. An Empirical Study on Robustness to Spurious Correlations using Pre-trained Language Models. *Transactions of the ACL*, 8:621–633.

Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. 2020. Towards Debiasing NLU Models from Unknown Biases. In *Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7597–7610, Online. ACL.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proc. of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the ACL*, pages 3544–3552, Online. ACL.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proc. of the 2020 Conf. EMNLP: System Demonstrations*, pages 38–45. ACL.

Mingzhu Wu, Nafise Sadat Moosavi, Andreas Rücklé, and Iryna Gurevych. 2020. Improving QA Generalization by Concurrent Modeling of Multiple Biases. *arXiv e-prints*, page arXiv:2010.03338.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised Data Augmentation. *CoRR*, abs/1904.12848v1.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the ACL*, pages 4334–4343, Florence, Italy. ACL.