YANMTT: Yet Another Neural Machine Translation Toolkit

Raj Dabre[1] **Diptesh Kanojia**[3]
Chinmay Sawant[2] **Eiichiro Sumita**[4]
National Institute of Information and Communications Technology[1,4]
Surrey Institute for People-centred AI[3]
University of Surrey[2,3]
[1]raj.dabre@nict.go.jp [2]chinmayssawant44@gmail.com
[3]d.kanojia@surrey.ac.uk [4]eiichiro.sumita@nict.go.jp

Abstract

In this paper, we present our open-source neural machine translation (NMT) toolkit called "Yet Another Neural Machine Translation Toolkit" abbreviated as YANMTT[1] which is built on top of the HuggingFace Transformers library. YANMTT aims to enable pre-training and fine-tuning of sequence-to-sequence models with ease. It can be used for training parameter-heavy models with minimal parameter sharing and efficient, lightweight models via heavy parameter sharing. Additionally, efficient fine-tuning can be done via fine-grained tuning parameter selection, adapter and prompt tuning. Our toolkit also comes with a user interface that can be used to demonstrate these models and visualize the attention and embedding representations. Apart from these core features, our toolkit also provides other advanced functionalities such as but not limited to document/multi-source NMT, simultaneous NMT, mixtures-of-experts and model compression.

1 Introduction

Neural machine translation (NMT) (Bahdanau et al., 2015) is an end-to-end machine translation (MT) approach known for obtaining state-of-the-art results for a variety of language pairs. Thanks to publicly available toolkits such as OpenNMT, Fairseq, Tensor2tensor, *etc.*; For NMT, and Natural Language Generation (NLG) in general, model training has become easier. Additionally, fine-tuning of large pre-trained models such as mBART (Liu et al., 2020) and mT5 (Xue et al., 2021) have led to significant advances for low-resource languages. Recently, the Transformers library from HuggingFace has made fine-tuning an accessible option. Parameter-efficient fine-tuning via adapters and prompts has recently gained popularity. This has led to the development of AdapterHub, which allows people to use pre-trained adapters for many tasks or easily train their custom adapters.

However, when it comes to pre-training sequence-to-sequence models from scratch, we noticed that there is very little support[2]. This is, presumably, because pre-training is computationally intensive, especially when the number of parameters is large, and not everyone can do this. However, not all situations need large models, and with the advancements in low-precision training and parameter-efficient architectures, we feel that democratizing pre-training is necessary. Additionally, relying on separate libraries for pre-training and fine-tuning can be quite exhausting. To this end, we decided to develop a publicly available toolkit to bring both into one place while simultaneously focusing on parameter efficiency.

We call our toolkit **YANMTT** which stands for "Yet Another Neural Machine Translation Toolkit" which is built on top of the Transformers library. YANMTT relies on a substantially modified version of the mBART model's code and contains simple scripts for NMT pre-training and fine-tuning. Our modifications revolve around parameter efficiency by implementing heavy parameter sharing and incorporating adapters and prompts into the model. In order to enable users to better understand, and modify if needed, the flow of pre-training and fine-tuning, we heavily annotate our code with comments. YANMTT also comes with a user interface that can be used to demo any developed models as well as analyze them by visualizing their attentions and embedding representations. We hope that YANMTT will help entice more researchers into advancing the field of efficient sequence-to-sequence pre-training and fine-tuning. While the main focus is on NMT, readers should note that this toolkit can also be used for general purpose NLG.

[1]https://github.com/prajdabre/yanmtt

[2]At the time of creating this toolkit in 2021, there was no easily available script for sequence-to-sequence pre-training.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 3: System Demonstrations, pages 257–263
July 10-12, 2023 ©2023 Association for Computational Linguistics

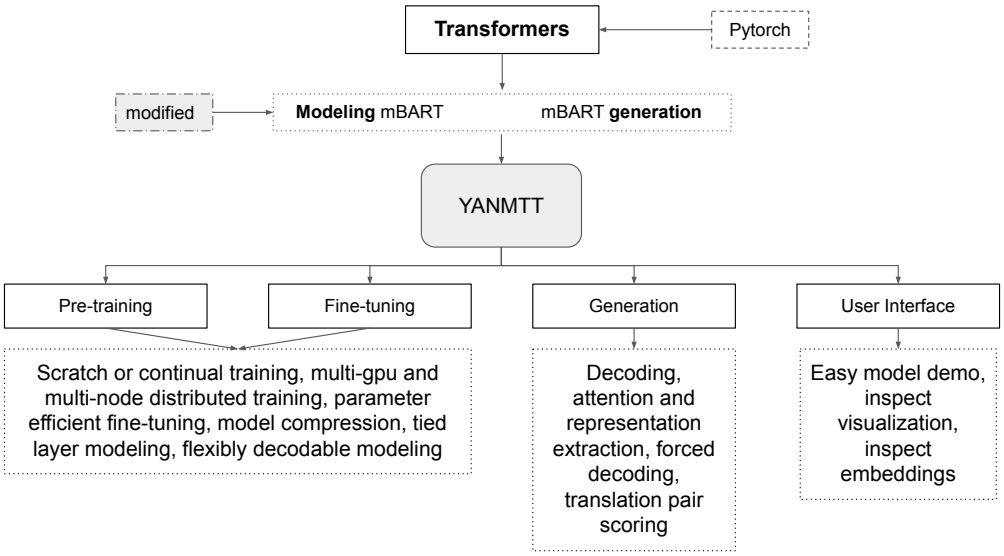

Figure 1: An architecture overview of the YANMTT.

2 Related Work

Tensor2tensor[3] is a deprecated library for training recurrent, convolutional as well as transformer models for a variety of sequence-to-sequence applications. It has been replaced by Trax[4]. While Tensor2tensor uses TensorFlow as a backend, Fairseq[5] is based on PyTorch and it also allows one to train a variety of NMT models. Unlike Tensor2tensor, Fairseq contains all necessary functionality for pre-training NMT models, but there is a severe lack of instructions for the same. OpenNMT (Klein et al., 2017), originally developed for recurrent NMT models, is based on TensorFlow as well as PyTorch. THUMT[6] is an NMT training toolkit based on TensorFlow, PyTorch and Theano.

Most recently, the Transformers (Wolf et al., 2020) library by HuggingFace, based on PyTorch and TensorFlow has become popular as it allows users to share trained models easily. In Transformers, the instructions for fine-tuning pre-trained models are abundant, but at the time of YANMTT's development (early 2021), there was no complete script for pre-training. On the HuggingFace hub, the central repository for all models trained with Transformers, it is possible to load and run models, but enabling users to locally demo and inspect their own models is also important from the perspective of privacy. Finally, for parameter-efficient fine-tuning, AdapterHub (Pfeiffer et al., 2020) builds on top of Transformers and enables users to leverage existing or custom-trained adapters.

All of the above toolkits and libraries are invaluable, but none appear to be a complete solution for sequence-to-sequence pre-training, parameter efficient fine-tuning and model demoing and inspection, a gap which YANMTT aims to fill.

3 The Toolkit: YANMTT

YANMTT, Yet Another Neural Machine Translation Toolkit, relies on the Transformers library and uses PyTorch. We use only the mBART implementation (for now) from Transformers and write several wrapper scripts enabling multilingual sequence-to-sequence pre-training, parameter efficient fine-tuning, decoding and attention and representation extraction. To enable users to quickly demonstrate and visually inspect trained models, we provide a user interface. We also modify the mBART modelling code to provide several advanced features. We provide the modified code along with our toolkit. We also provide example data and usage instructions in the form of example scripts. We encourage the reader to look at our toolkit[7] and watch the demo video[8]. Figure 1 contains an overview of YANMTT architecture.

[3] https://github.com/tensorflow/tensor2tensor
[4] https://github.com/google/trax
[5] https://github.com/facebookresearch/fairseq
[6] https://github.com/THUNLP-MT/THUMT
[7] https://github.com/prajdabre/yanmtt
[8] https://youtu.be/ee38gda5qnc

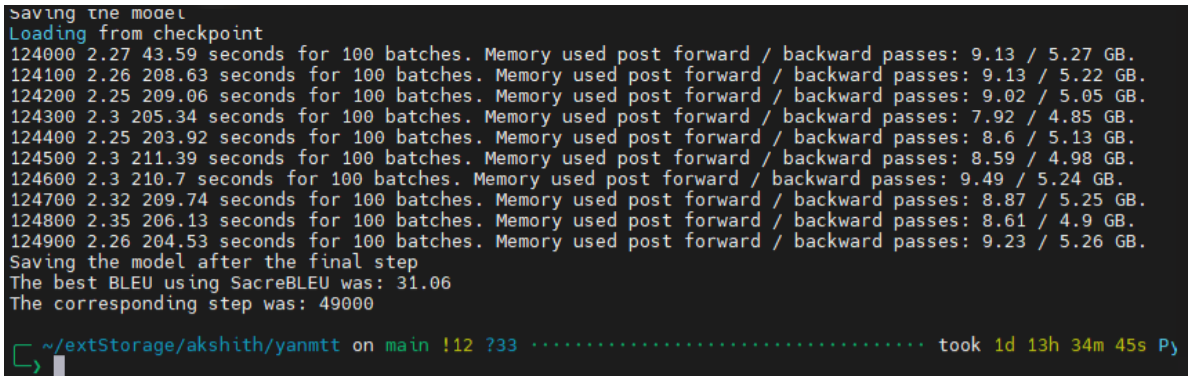

Figure 2: Screenshot obtained after fine-tuning mBART-50 for English-Telugu NMT where the model training was performed over 1 x A40 GPU over ∼1 day and 13 hours.

3.1 Training

YANMTT enables multilingual training of sequence-to-sequence models; while also supporting full or mixed-precision training on a single GPU or on multiple GPUs[9] across machines.

Tokenization: While mBART uses BPE, mT5 uses *sentencepiece* for subword segmentation. YANMTT allows users to train and use both types of subword segmenters.

Pre-training: We currently support mBART style text-infilling and sentence shuffling, as well as mT5 and MASS style masked span-prediction for pre-training from scratch. Note that sentence shuffling is optional since it is useful only when the pre-training data is in the form of documents. We also support continued pre-training of existing mBART-25, mBART-50, BART and IndicBART (Dabre et al., 2022) models[10]. This should enable users to adapt pre-trained models using monolingual corpora on a downstream NLG task such as NMT.

Fine-tuning: Users may train their own sequence-to-sequence models from scratch or fine-tune pre-trained models. Fine-tuning can be done on the user's own models or official pre-trained models like mBART-25, mBART-50, BART and IndicBART. Users have fine-grained control over what parts of the pre-trained models they want to use for partial initialization. This way, the fine-tuned[11] models can be shrunk or grown as required.

Logging: Figure 2 shows the logs generated during model training depicting model loss, evaluation set scores, timings and memory consumption. Although we do not show it here, this, along with gradient information, is also logged on TensorBoard.

3.2 Parameter Efficiency

Using YANMTT, parameter efficiency can be achieved when fine-tuning or training from scratch.

Lightweight Fine-Tuning: We enable fine-tuning of adapters and prompts to enable users to fine-tune minimal number of parameters. We implement various adapters mentioned in He et al. (2022) such as Houlsby adapter, FFN-only adapter, parallel adapter, hypercomplex adapter (Le et al., 2021) and IA^3 adapter (Liu et al., 2022). Regarding prompts, currently prefix tuning is supported. It is also possible to mix-and-match adapters and prompts. Users may also specify a list of parameters which they want to fine-tune for more control over the process.

Parameter Sharing: When training models from scratch, we enable recurrently stacked layers (Dabre and Fujita, 2019) involving tying parameters of layers and the resultant models, which we call ALBERT, tend to be 50-70% smaller.

3.3 Decoding

Users can decode the models, extract encoder or decoder representations, generate heatmaps of attentions and score translation pairs.

3.4 User Interface (UI)

To enable users to locally test or demo their models and inspect them, we provide a web-based UI based on Flask. This interface can be hosted online by using software such as ngrok[12] or an Apache server.

[9] We directly use the distributed data-parallel functionality of PyTorch instead of the Accelerate library so users better can understand and control how distribution is done.

[10] Any models based on the BART or mBART architecture will work. We plan to support the adaptation of other sequence-to-sequence models soon.

[11] not just for fine-tuning on a downstream task, but also when doing continued pre-training. Additionally, this functionality may be used if one wishes to expand the model's capacity step by step.

[12] https://ngrok.com

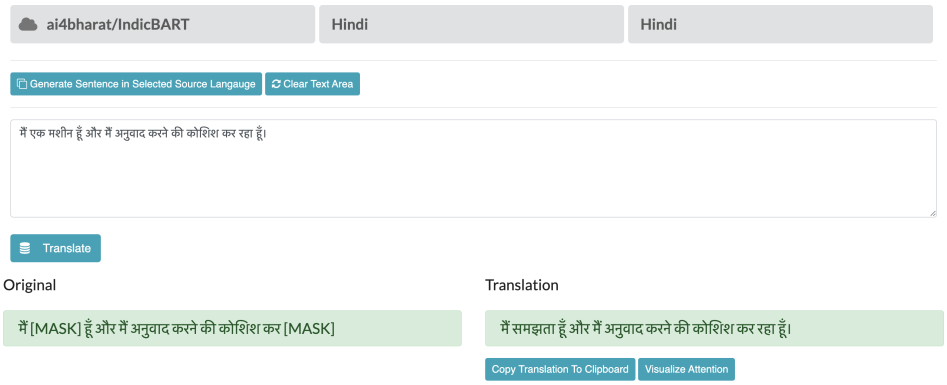

Figure 3: YANMTT User Interface for inspecting the IndicBART model by masking random words from the input sentence in the Hindi language. IndicBART was trained using YANMTT.

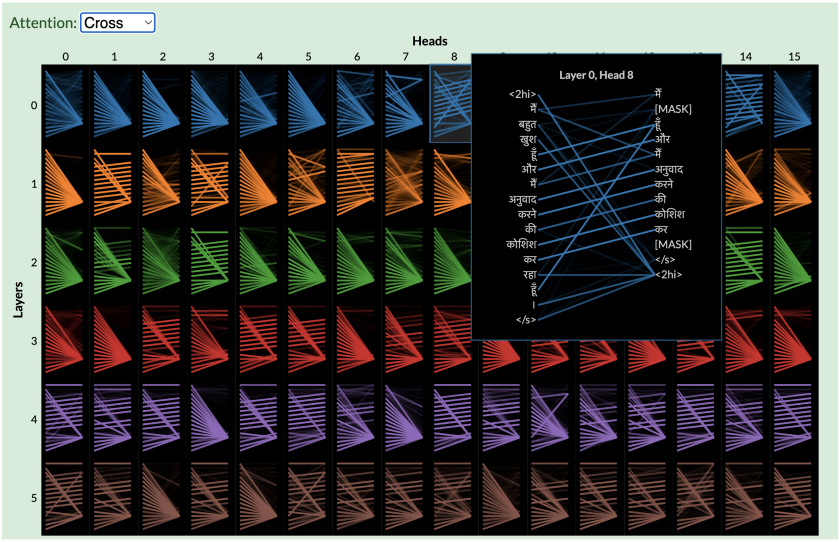

Figure 4: Visualization of cross-attention, where masked words are predicted for a sentence in Hindi.

Model Decoding: Using the GUI, the user may load models and decode sentences. Models supported are those trained locally or official based on the mBART code. If the model to be used is a denoising model, then users may use decoding for MASK prediction as in Figure 3. If the model to be used was fine-tuned for a downstream task such as NMT, then it can be used for generating relevant outputs for that task input.

Attention Visualization: To enable users to inspect their model attentions, we use the bertviz[13] library to visualize multi-head, self- or cross-attention across all layers in the encoder and decoder. As shown in Figure 4, users can select the type of attention, the desired layer and attention head and get more detailed information.

Representation Visualization: We also integrate TensorBoard embedding projector visualization into our user interface to visualize the model layer representations[14] as shown in Figure 5. This can be especially useful for understanding model representations in multilingual settings.

3.5 Advanced Features

In addition to the core features above, we also enable YANMTT users to perform the following:

Model Compression: We enable users to compress models by distillation (Kim and Rush, 2016) of models using either or all of 3 different ways: teacher-student cross-entropy minimization , minimizing the mean squared difference between the hidden layer representations of the teachers and students and minimizing the cross entropy between the self/cross-attentions of the teacher and students.

[13] https://github.com/jessevig/bertviz

[14] For now, we support the encoder's final layer's average representation. Additional flexibility in terms of layer choice as well as decoder representations will be provided in future versions of YANMTT.

Figure 5: Sentence representations visualization via TensorBoard integrated into the YANMTT user interface.

Mixtures-of-Experts: Mixtures-of-expert (MoE) layers can replace standard feed-forward layers and enable the user to train large models with a billion parameters or more. Model parallel training is currently unavailable, so the largest trainable model is limited by computing resources available[15]. A similar approach was used by Facebook to train the largest known multilingual NMT model supporting over 200 language pairs (Costa-jussà et al., 2022).

Document and Multi-source Translation: Pre-trained models often use document-level data, and we considered it prudent to support explicit document-level translation approaches. Since document context is often treated as an additional input, our document translation implementation can be used for multi-source MT (Dabre et al., 2017).

Simultaneous Translation: In a real-time setting, simultaneous MT can be useful. To enable users to test their models in simultaneous MT settings, we implement 'wait-k' training and decoding (Ma et al., 2019) which can also be combined with document and multi-source MT. Apart from this, there are several **other features which are listed on the YANMTT's GitHub** repository.

3.6 YANMTT Adoption & Future Plans

YANMTT has 93 stars on GitHub, indicating that it is being noticed and used. In particular, IndicBART[16], its variants and fine-tuned version (Dabre et al., 2022) were developed with YANMTT and have seen around 8500 downloads thus far from HuggingFace hub. We have the following future plans for our toolkit:

1. Supporting additional pre-training approaches like PEGASUS and CSP.
2. Low-precision model parallel training for better scaling of large models.
3. Comprehensive support for all types of adapters and prompt tuning.
4. An improved user interface to enable better visualization of model internals.
5. Integration of existing post-hoc model explainability techniques.

4 Conclusion

We have presented our open-source toolkit called "Yet Another Neural Machine Translation Toolkit", also known as YANMTT. YANMTT allows users to pre-train and fine-tune their own multilingual sequence to sequence models. Our toolkit can be used for training models at a reasonable scale, as well as to perform parameter efficient fine-tuning via adapters and prompts. We provide a convenient user interface for model demonstration and inspection, as well as the ability to visualize attention and model representations. We also implemented functionalities for compressing large models via selective parameter transfer and knowledge distillation approaches. Additionally, we have provided basic functionalities for simultaneous and document/multi-source NMT. YANMTT appears to be modestly adopted by researchers, and we plan to further specialize it for better at-scale training and efficient fine-tuning.

[15] With A100 80 GB GPUs, it is possible to train models with 10-20 billion parameters given reasonable batch size.

[16] https://huggingface.co/ai4bharat/IndicBART

5 Limitations

Although YANMTT can be used to train models at scale, the lack of model-parallel training limits the size of models to those that can fit on single GPUs. YANMTT tokenizes sentences on the fly and does not pre-process them, so the overall training speed is slightly slower than that of Fairseq or Tensor2tensor; however, we plan to fix this soon.

6 Acknowledgements

A part of this work was conducted under the commissioned research program "Research and Development of Advanced Multilingual Translation Technology" in the "R&D Project for Information and Communications Technology (JPMI00316)" of the Ministry of Internal Affairs and Communications (MIC), Japan.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

Raj Dabre, Fabien Cromieres, and Sadao Kurohashi. 2017. Enabling multi-source neural machine translation by concatenating source sentences in multiple languages. In *Proceedings of MT Summit XVI, vol.1: Research Track*, pages 96–106, Nagoya, Japan.

Raj Dabre and Atsushi Fujita. 2019. Recurrent stacking of layers for compact neural machine translation models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6292–6299.

Raj Dabre, Himani Shrotriya, Anoop Kunchukuttan, Ratish Puduppully, Mitesh Khapra, and Pratyush Kumar. 2022. IndicBART: A pre-trained model for indic natural language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1849–1863, Dublin, Ireland. Association for Computational Linguistics.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Tuan Le, Marco Bertolini, Frank Noé, and Djork-Arné Clevert. 2021. Parameterized hypercomplex graph neural networks for graph classification. In *Artificial Neural Networks and Machine Learning – ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part III*, page 204–216, Berlin, Heidelberg. Springer-Verlag.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Mingbo Ma, Liang Huang, Hao Xiong, Renjie Zheng, Kaibo Liu, Baigong Zheng, Chuanqiang Zhang, Zhongjun He, Hairong Liu, Xing Li, Hua Wu, and Haifeng Wang. 2019. STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.