THUMT: An Open-Source Toolkit for Neural Machine Translation

Zhixing Tan[†], **Jiacheng Zhang**[†], **Xuancheng Huang**[†], **Gang Chen**[†], **Shuo Wang**[†], **Maosong Sun**[†‡], **Huanbo Luan**[†], and **Yang Liu**[†‡] [*]
[†]Institute for Artificial Intelligence
Department of Computer Science and Technology, Tsinghua University
Beijing National Research Center for Information Science and Technology
[‡]Beijing Academy of Artificial Intelligence

Abstract

THUMT is an open-source toolkit for neural machine translation (NMT) developed by the Natural Language Processing Group at Tsinghua University. The toolkit is easy to use, modify and extend while provides the latest advances in NMT research and production. THUMT implements several standard NMT models and supports distributed training across multiple machines, fast inference, and model visualization. Experiments on English-German and Chinese-English datasets show that THUMT can obtain results that are comparable to state-of-the-art NMT systems.

1 Introduction

Machine translation (MT), which investigates the use of computers to translate human languages automatically, is an important task in natural language processing and artificial intelligence communities. With the availability of bilingual machine-readable texts, data-driven approaches to machine translation have gained wide popularity since the 1990s (Hutchins and Lovtskii, 2000). Recent several years have witnessed the rapid development of end-to-end neural machine translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2014; Vaswani et al., 2017). Capable of learning representations from data, NMT has quickly replaced conventional statistical machine translation (SMT) (Brown et al., 1993; Koehn et al., 2003; Chiang, 2005) to become the new *de facto* method in practical MT systems (Wu et al., 2016).

This paper introduces THUMT, an open-source NMT toolkit targeting both academia and industry. THUMT originally developed with Theano (Theano Development Team, 2016) and begins its launch in June 2017. With the emerging of new deep learning frameworks, THUMT added TensorFlow (Abadi et al., 2016) implementation in October 2017 and PyTorch (Paszke et al., 2019) implementation in August 2019. The current status of the three implementations are as follows:

- THUMT-Theano (Zhang et al., 2017): the original project developed with Theano, which is no longer updated because MLA put an end to Theano. It implemented the standard attention-based model (RNNsearch) (Bahdanau et al., 2014), minimum risk training (MRT) (Shen et al., 2015) for optimizing model parameters with respect to evaluation metrics, semi-supervised training (SST) (Cheng et al., 2016) for exploiting monolingual

[*]Corresponding author.

Features	Theano	TensorFlow	PyTorch
Models	RNNsearch	Seq2Seq, RNNsearch, Transformer	Transformer
Criterions	MLE, MRT, SST	MLE	MLE
Optimizers	SGD, Adadelta, Adam	Adam	SGD, Adadelta, Adam
LRP	Yes	Yes	No
Gradient Aggregation	No	Yes	Yes
Distributed Training	No	Yes	Yes
Mixed-Precision	No	Yes	Yes
TensorBoard	No	Yes	Yes

Table 1: Available features in different implementations.

corpora to learn bi-directional translation models, and layer-wise relevance propagation (LRP) (Ding et al., 2017) for visualizing and analyzing RNNsearch.

- THUMT-TensorFlow: an implementation focuses on performance. It implemented the sequence-to-sequence model (Seq2Seq) (Sutskever et al., 2014), the standard attention-based model (RNNsearch) (Bahdanau et al., 2014), the Transformer model (Transformer) (Vaswani et al., 2017), and LRP visualization for RNNsearch and Transformer. It also added new features such as multi-GPU training, distributed training, ensemble inference, and TensorBoard visualization.

- THUMT-PyTorch: a new implementation developed with PyTorch, which is more flexible and easier to use by the virtue of dynamic eager execution. It implemented the Transformer model and also supports multi-GPU training, distributed training, ensemble inference, and TensorBoard visualization.

THUMT is developed by the Tsinghua Natural Language Processing Group. The latest source code is available at GitHub [1] and is dual licensed. Open-source licensing is under the BSD-3-Clause, which allows free use for research purposes. THUMT has been used in many researches as well as several production MT systems.

2 Features

The primary goal of THUMT is to provide a toolkit that is easy to run and modify while featuring the latest deep learning techniques. The design of THUMT is highly modular. It is easy to add new models, optimizers, and learning rate schedules to THUMT. The toolkit provides command-line interface to train and infer from an NMT model, and also supports multi-GPU training, distributed training as well as mixed-precision training to make full use of the modern hardware features. Table 1 lists available features in different implementations. We will give a brief introduction to the components and features provided by the toolkit.

2.1 Models

THUMT implemented three mainstream NMT models: Seq2Seq (Sutskever et al., 2014), RNNsearch (Bahdanau et al., 2014), and Transformer (Vaswani et al., 2017). Seq2Seq uses a recurrent neural network (RNN) to encode the input sentence into a fixed-size hidden representation and uses another RNN to generate translation conditioned on the representation. RNNsearch exploits variable representation with attention mechanism and achieves significant improvements over Seq2Seq model. Transformer uses deep self-attention layers instead of RNN layers in both encoder and decoder. It achieves the best performance among the three models.

[1] `https://github.com/THUNLP-MT/THUMT`

THUMT also implemented relative position embedding (Shaw et al., 2018) in its Transformer implementation. Instead of adding absolute positions to its input, this approach considers relative positions in the self-attention mechanism. We offer two options in our Tensorflow implementation, one can turn on or off the relative position embedding and set the desired maximum relative distance. Note that we only implement relative position embedding in self-attention but not in encoder-decoder attention.

2.2 Training

THUMT supports both single machine multi-GPU training and multi-machine distributed training. THUMT-TensorFlow uses the Horovod (Sergeev and Del Balso, 2018) toolkit for distributed training while THUMT-PyTorch adopts `torch.distributed` module to achieve the same functionality.

Modern NMT models (e.g. Transformer) usually require using large batch sizes during training. However, it is normally impractical to fit a large batch into a single GPU device. To alleviate this problem, we allow THUMT to split the batch into several smaller batches and collect the gradient on each batch independently. Then we aggregate the gradients and perform optimization. Gradient aggregation simulates large batch size training with multiple small batch training, which significantly reduce the memory requirement for training NMT models.

The latest GPU (e.g. Nvidia V100) supports half-precision computation which significantly improve the training speed and reduce the memory requirement. However, training with half-precision is less stable than single-precision because of the reduced precision. To address this problem, THUMT implemented mixed-precision training which uses half-precision in the forward pass while switches to single-precision in the backward pass to maintain numerical stability. Furthermore, THUMT also apply dynamic loss scaling during the forward pass to increase the numerical precision.

THUMT provides different optimizers, learning rate schedules, and validation functionality for users to control the training process. These options can be easily changed through the command-line interface. New optimizers and learning rate schedules can be easily integrated into THUMT with minor modifications.

2.3 Inference

THUMT supports beam-search and random sampling during inference. The user can specify the batch size, beam size, and length penalty before decoding. Apart from beam-search, the user can also choose to employ random sampling, which samples a word from multinomial distribution at each step. THUMT also supports decoding with half-precision to speed up the inference stage.

Model averaging is beneficial to the performance of neural machine translation models. THUMT provides an additional script to carry out model averaging. The users can either average the lastest n checkpoints or the top-n best checkpoints sorted by validation scores into a single checkpoint.

Model ensemble is another way to improve the performance of neural machine translation. THUMT can ensemble multiple NMT models regardless of their architecture, provided all models share the same vocabulary. When performing inference, if the number of checkpoints assigned by the users is more than one, THUMT will perform model ensemble automatically by calculating an arithmetic mean of the log-probabilities provided by all checkpoints. Normally, model ensemble can achieve significant improvements. However, it requires more memory and computations.

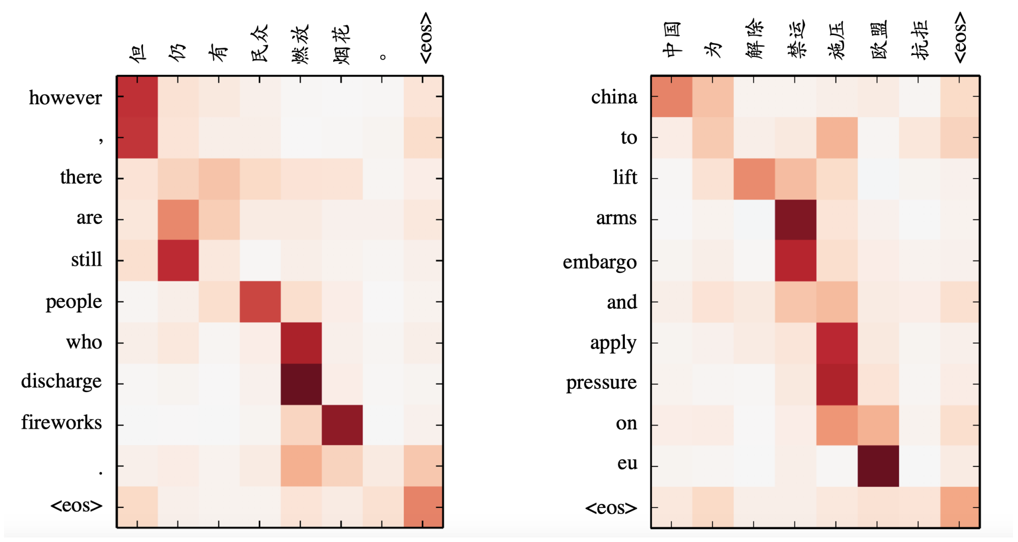

Figure 1: Visualization with LRP for the Transformer Model.

2.4 Visualization

Although NMT achieves state-of-the-art translation performance, it is hard to understand how it works because all internal information is represented as real-valued vectors or matrices. To address this problem, THUMT features a visualization tool to use layer-wise relevance propagation (LRP) (Bach et al., 2015) to visualize and interpret neural machine translation models (Ding et al., 2017).

Figure 1 shows an example of visualizing the Transformer model. Although the transformer model uses attention mechanisms extensively, it is hard to determine the most representative head. For LRP, it is possible to calculate the global relevance between source words and target words, which helps analyze the internal workings of NMT. Please refer to Ding et al. (2017) for more details.

3 Experiments

3.1 Setup

We evaluate THUMT on English-German and Chinese-English translation tasks. The evaluation metric is case-sensitive BLEU (Papineni et al., 2002). Following Vaswani et al. (2017), translations are generated via beam-search with a beam size of 4 and a length penalty of 0.6.

For English-German, we use the WMT14 training corpus which contains 4.5M sentence pairs with 103M English words and 96M German words. We also use a shared source-target vocabulary of about 37000 tokens encoded by BPE (Sennrich et al., 2016). We use `newstest2014` as the test set.

For Chinese-English translation, we use the training corpus provided by WMT18 [2]. The corpus consists of 24M sentence pairs with 509M Chinese words and 576M English words. We use 32K BPE operations to build vocabularies. The test set is `newstest2017`.

We train the Transformer model on the two datasets. Unless otherwise noted, the setting is the same as Vaswani et al. (2017). All models are trained on 4 machines interconnected with InfiniBand, and each machine has 8 GTX 2080Ti GPUs.

[2]`http://data.statmt.org/wmt18/translation-task/preprocessed/zh-en`

Model	Size	Speed (tokens/sec)	
		Training	*Inference*
Transformer	Base	60K	620
Transformer	Big	35K	550

Table 2: Speed of THUMT, evaluated on WMT14 English-German (En-De). Speed of training is reported with a 8-GPU setting and speed of inference is reported with a single GPU setting.

Direction	Model	Size	Steps	GPUs	BS/GPU	Precision	BLEU
En → De	Transformer (Vaswani et al., 2017)	Base	100K	8	-	FP32	27.30
		Big	300K	8	-	FP32	28.40
Zh → En	Transformer (Hassan et al., 2018)	Big	-	-	-	FP32	24.20
En → De	Transformer	Base	100K	4	2×4096	FP16	26.85
		Base	100K	8	4096	FP16	26.76
		Base	100K	4	2×4096	FP32	26.91
		Base	100K	8	4096	FP32	26.95
		Base	85K	8	2×4096	FP16	27.40
		Big	300K	8	4096	FP16	28.71
		Big	20K	16	8×4096	FP16	28.68
		Big	25K	32	8×2048	FP16	28.51
Zh → En	Transformer	Big	300K	8	2×4096	FP16	24.07

Table 3: Benchmarks on WMT14 English-German (En-De) and WMT18 Chinese-English (Zh-En) datasets. We use "2×4096" to denote aggregating gradients for 2 steps with 4096 batch size per step. "FP32" and "FP16" denote training with single-precision and mixed-precision, respectively. We use distributed settings enable training with more than 8 GPUs.

3.2 Results

Table 2 shows the speed of THUMT. The speed of training is reported with a 8-GPU setting and the speed of inference is reported with a single GPU setting. The training speed of Transformer model is around 35K to 60K tokens per second, depending on the model size. It takes about 1 day to train a Transformer big model with 32 GPUs. The inference speed is about 620 tokens per second for the base Transformer model and 550 tokens per second for the big model. Currently, THUMT does not support inference with CPUs, and we plan to add this functionality in the future.

Table 3 shows the results on English-German and Chinese-English translation. For English-German, we trained Transformer base/big models with several different settings by varying model sizes, training steps, and number of GPUs. All Transformer big models performed better than Transformer base models. When we trained the Transformer base model using mixed-precision (FP16) instead of single-precision (FP32), the performance only drops slightly. As the number of tokens in each mini-batch increases from 32,768 to 65,536, the performance improves from 26.95 to 27.40 even though the training steps reduce from 100K to 85K. For Chinese-English, we trained a Transformer big model using mixed-precision in 300K steps, and the number of tokens in mini-batch is 65,536. The BLEU score on the test dataset is 24.07. The models trained using THUMT are comparable to those reported by Vaswani et al. (2017) and Hassan et al. (2018) in terms of BLEU score.

4 Conclusion and Future works

We have introduced a new open-source toolkit for NMT that supports mainstream models, distributed training, and fast inference. The toolkit also features a visualization tool for analyzing the translation process of THUMT. The toolkit is freely available at `http://thumt.thunlp.org`.

Currently, the users still rely on external tools to prepare the training corpus. We plan to add preprocessing functionality to make THUMT self-contained. We will continually add new features to THUMT to make it a better toolkit for both research and production.

Acknowledgements

This work was supported by the National Key R&D Program of China (No. 2017YFB0202204), National Natural Science Foundation of China (No. 61925601, No. 61761166008, No. 61772302), Beijing Academy of Artificial Intelligence, and the NExT++ project supported by the National Research Foundation, Prime Ministers Office, Singapore under its IRC@Singapore Funding Initiative.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *Proceedings of OSDI*.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*.

Cheng, Y., Xu, W., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2016). Semi-supervised learning for neural machine translation. *arXiv preprint arXiv:1606.04596*.

Chiang, D. (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*.

Ding, Y., Liu, Y., Luan, H., and Sun, M. (2017). Visualizing and understanding neural machine translation. In *Proceedings of ACL*.

Hassan, H., Aue, A., Chen, C., Chowdhary, V., Clark, J., Federmann, C., Huang, X., Junczys-Dowmunt, M., Lewis, W., Li, M., et al. (2018). Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*.

Hutchins, J. and Lovtskii, E. (2000). Petr petrovich troyanskii (1894–1950): A forgotten pioneer of mechanical translation. *Machine translation*, 15(3):187–221.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of NAACL*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2002). Bleu: A method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of NIPS*.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of ACL*.

Sergeev, A. and Del Balso, M. (2018). Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. In *Proceedings of NAACL-HLT*.

Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2015). Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of NIPS*.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhang, J., Ding, Y., Shen, S., Cheng, Y., Sun, M., Luan, H., and Liu, Y. (2017). Thumt: An open source toolkit for neural machine translation. *arXiv preprint arXiv:1706.06415*.