OpenNMT: Open-Source Toolkit for Neural Machine Translation

Guillaume Klein[†], Yoon Kim[*], Yuntian Deng[*], Jean Senellart[†], Alexander M. Rush[*]
Harvard University[*], SYSTRAN [†]

Abstract

We describe an open-source toolkit for neural machine translation (NMT). The toolkit prioritizes efficiency, modularity, and extensibility with the goal of supporting NMT research into model architectures, feature representations, and source modalities, while maintaining competitive performance and reasonable training requirements. The toolkit consists of modeling and translation support, as well as detailed pedagogical documentation about the underlying techniques.

1 Introduction

Neural machine translation (NMT) is a new methodology for machine translation that has led to remarkable improvements, particularly in terms of human evaluation, compared to rule-based and statistical machine translation (SMT) systems (Wu et al., 2016; Crego et al., 2016). Originally developed using pure sequence-to-sequence models (Sutskever et al., 2014; Cho et al., 2014) and improved upon using attention-based variants (Bahdanau et al., 2014; Luong et al., 2015), NMT has now become a widely-applied technique for machine translation, as well as an effective approach for other related NLP tasks such as dialogue, parsing, and summarization.

As NMT approaches are standardized, it becomes more important for the machine translation and NLP community to develop open implementations for researchers to benchmark against, learn from, and extend upon. Just as the SMT community benefited greatly from toolkits like Moses (Koehn et al., 2007) for phrase-based SMT and CDec (Dyer et al., 2010) for syntax-based SMT, NMT toolkits can provide a foundation to build upon. A toolkit should aim to provide a shared framework for developing and comparing open-source systems, while at the same time being efficient and accurate enough to be used in production contexts.

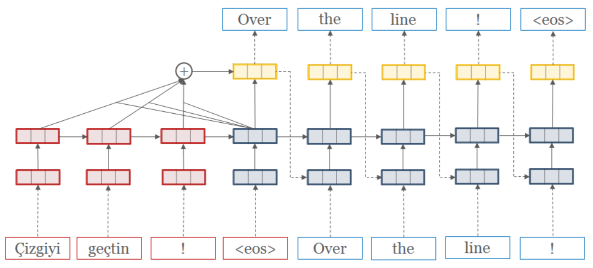


Figure 1: Schematic view of neural machine translation. The red source words are first mapped to word vectors and then fed into a recurrent neural network (RNN). Upon seeing the ⟨eos⟩ symbol, the final time step initializes a target blue RNN. At each target time step, *attention* is applied over the source RNN and combined with the current hidden state to produce a prediction $p(w_t | w_{1:t-1}, x)$ of the next word. This prediction is then fed back into the target RNN.

Currently there are several existing NMT implementations. Many systems such as those developed in industry by Google, Microsoft, and Baidu, are closed source, and are unlikely to be released with unrestricted licenses. Many other systems such as *GroundHog*, *Blocks*, *tensorflow-seq2seq*, *lamtram*, and our own *seq2seq-attn*, exist mostly as research code. These libraries provide important functionality but minimal support to production users. Perhaps most promising is the University of Edinburgh's *Nematus* system originally based on NYU's NMT system. Nematus provides high-accuracy translation, many options, clear documentation, and has been used in several successful research projects. In the development of this project, we aimed to build upon the strengths of this system, while providing additional documentation and functionality to provide a useful open-source NMT framework for the NLP community in academia and industry.

With these goals in mind, we introduce *OpenNMT* (http://opennmt.net), an open-source framework for neural machine translation. OpenNMT is a complete NMT implementation. In addition to providing code for the core translation tasks, OpenNMT was designed with three aims: (a) prioritize first training and test efficiency, (b) maintain model modularity and readability, (c) support significant research extensibility.

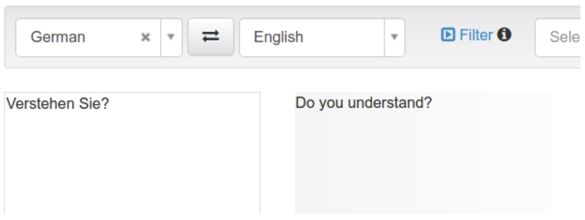
This engineering report describes how the system targets these criteria. We begin by briefly surveying the background for NMT, describing the high-level implementation details, and then describing specific case studies for the three criteria. We end by showing benchmarks of the system in terms of accuracy, speed, and memory usage for several translation and translation-like tasks.

2 Background

NMT has now been extensively described in many excellent tutorials (see for instance https://sites.google.com/site/acl16nmt/home). We give only a condensed overview.

NMT takes a conditional language modeling view of translation by modeling the probability of a target sentence $w_{1:T}$ given a source sentence $x_{1:S}$ as $p(w_{1:T}|x) = \prod_1^T p(w_t|w_{1:t-1}, x; \theta)$. This distribution is estimated using an attention-based encoder-decoder architecture (Bahdanau et al., 2014). A source encoder recurrent neural network (RNN) maps each source word to a word vector, and processes these to a sequence of hidden vectors $\mathbf{h}_1, \ldots, \mathbf{h}_S$. The target decoder combines an RNN hidden representation of previously generated words $(w_1, ...w_{t-1})$ with source hidden vectors to predict scores for each possible next word. A softmax layer is then used to produce a next-word distribution $p(w_t|w_{1:t-1}, x; \theta)$. The source hidden vectors influence the distribution through an attention pooling layer that weights each source word relative to its expected contribution to the target prediction. The complete model is trained end-to-end to minimize the negative log-likelihood of the training corpus. An unfolded network diagram is shown in Figure 1.

In practice, there are also many other important aspects that improve the effectiveness of the base model. Here we briefly mention four areas: (a) It is important to use a gated RNN such as



Verstehen Sie?

Do you understand?

Figure 2: Live demo of the OpenNMT system across dozens of language pairs.

an LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Chung et al., 2014) which help the model learn long-term features. (b) Translation requires relatively large, stacked RNNs, which consist of several vertical layers (2-16) of RNNs at each time step (Sutskever et al., 2014). (c) Input feeding, where the previous attention vector is fed back into the input as well as the predicted word, has been shown to be quite helpful for machine translation (Luong et al., 2015). (d) Test-time decoding is done through *beam search* where multiple hypothesis target predictions are considered at each time step. Implementing these correctly can be difficult, which motivates their inclusion in an NMT framework.

3 Implementation

OpenNMT is a complete library for training and deploying neural machine translation models. The system is successor to *seq2seq-attn* developed at Harvard, and has been completely rewritten for ease of efficiency, readability, and generalizability. It includes vanilla NMT models along with support for attention, gating, stacking, input feeding, regularization, beam search and all other options necessary for state-of-the-art performance.

The main system is implemented in the Lua/Torch mathematical framework, and can be easily be extended using Torch's internal standard neural network components. It has also been extended by Adam Lerer of Facebook Research to support Python/PyTorch framework, with the same API.

The system has been developed completely in the open on GitHub at (http://github.com/opennmt/opennmt) and is MIT licensed. The first version has primarily (intercontinental) contributions from SYSTRAN Paris and the Harvard NLP group. Since official beta release, the project has been starred by over 1000 users, and there have been active development by those outside of these two organizations. The project has an active forum for community feedback with over five hundred posts in the last two months. There is also a live demonstration available of the system in use (Figure 3).

One nice aspect of NMT as a model is its relative compactness. The Lua OpenNMT system including preprocessing is roughly 4K lines of code, and the Python version is less than 1K lines (although slightly less-feature complete). For comparison the Moses SMT framework including language modeling is over 100K lines. This makes the system easy to completely understand for newcomers. The project is fully self-contained depending on minimal number of external Lua libraries and including also a simple language independent reversible tokenization and detokenization tools.

4 Design Goals

As the low-level details of NMT have been covered previously, we focus this report on the design goals of OpenNMT: system efficiency, code modularity, and model extensibility.

4.1 System Efficiency

As NMT systems can take from days to weeks to train, training efficiency is a paramount concern. Slightly faster training can make be the difference between plausible and impossible experiments.

Memory Sharing When training GPU-based NMT models, memory size restrictions are the most common limiter of batch size, and thus directly impact training time. Neural network toolkits, such as Torch, are often designed to trade-off extra memory allocations for speed and declarative simplicity. For OpenNMT, we wanted to have it both ways, and so we implemented an external memory sharing system that exploits the known time-series control flow of NMT systems and aggressively shares the internal buffers between clones. The potential shared buffers are dynamically calculated by exploration of the network graph before starting training. In practical use, aggressive memory reuse provides a saving of 70% of GPU memory with the default model size.

Multi-GPU OpenNMT additionally supports multi-GPU training using data parallelism. Each GPU has a replica of the master parameters and process independent batches during training phase. Two modes are available: synchronous and asynchronous training. In synchronous training, batches on parallel GPU are run simultaneously and gradients aggregated to update master parameters before resynchronization on each GPU for the following batch. In asynchronous training, batches are run independent on each GPU, and independent gradients accumulated to the master copy of the parameters. Asynchronous SGD is known to provide faster convergence (Dean et al., 2012). Experiments with 8 GPUs show a 6× speed up in per epoch, but a slight loss in training efficiency. When training to similar loss, it gives a 3.5× total speed-up to training.

Batch	Beam	GPU	CPU	CPU/C
1	5	209.0	24.1	62.2
1	1	166.9	23.3	84.9
30	5	646.8	104.0	116.2
30	1	535.1	128.5	392.7

Table 1: Performance numbers in source tokens per second for the Torch CPU/GPU implementations and for the multithreaded CPU C implementation. (Run with Intel i7/GTX 1080)

C/Mobile/GPU Translation Training NMT systems requires significant code complexity to facilitate fast back-propagation-through-time. At deployment, the system is much less complex, and only requires (i) forwarding values through the network and (ii) running a beam search that is much simplified compared to SMT. OpenNMT includes several different translation deployments specialized for different run-time environments: a batched CPU/GPU implementation for very quickly translating a large set of sentences, a simple single-instance implementation for use on mobile devices, and a specialized C implementation. The first implementation is suited for research use, for instance allowing the user to easily include constraints on the feasible set of sentences and ideas such as pointer networks and copy mechanisms. The last implementation is particularly suited for industrial use as it can run on CPU in standard production environments; it reads the structure of the network and then uses the *Eigen* package to implement the basic linear algebra necessary for decoding. Table 4.1 compares the performance of the different implementations based on batch size, beam size.

4.2 Modularity for Research

A secondary goal was a desire for code readability for non-experts. We targeted this goal by explicitly separating out many optimizations from the core model, and by including tutorial documentation within the code. To test whether this approach would allow novel feature development we experimented with two case studies.

Case Study: Factored Neural Translation In feature-based factored neural translation (Sennrich and Haddow, 2016), instead of generating a word at each time step, the model generates both word and associated features. For instance, the system might include words and separate case features. This extension requires modifying both the inputs and the output of the decoder to generate multiple symbols. In OpenNMT both of these aspects are abstracted from the core translation code, and therefore factored translation simply modifies the input network to instead process the feature-based representation, and the output generator network to instead produce multiple conditionally independent predictions.

Case Study: Attention Networks The use of attention over the encoder at each step of translation is crucial for the model to perform well. The default method is to utilize the global attention mechanism. However there are many other types of attention that have recently proposed including local attention (Luong et al., 2015), sparse-max attention (Martins and Astudillo, 2016), hierarchical attention (Yang et al., 2016) among others. As this is simply a module in OpenNMT it can easily be substituted. Recently the Harvard group developed a *structured* attention approach, that utilizes graphical model inference to compute this attention. The method is quite computationally complex; however as it is modularized by the Torch interface, it can be used in OpenNMT to substitute for standard attention.

4.3 Extensibility

Deep learning is a quickly evolving field. Recently work such as variational seq2seq auto-encoders (Bowman et al., 2016) or memory networks (Weston et al., 2014), propose interesting extensions to basic seq2seq models. We next discuss a case study to demonstrate that OpenNMT is extensible to future variants.

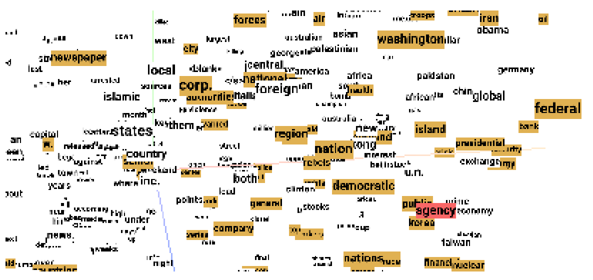


Figure 3: 3D Visualization of OpenNMT source embedding from the TensorBoard visualization system.

Multiple Modalities Recent work has shown that NMT-like systems are effective for image-to-text generation tasks (Xu et al., 2015). This task is quite different from standard machine translation as the source sentence is now an image. However, the future of translation may require this style of (multi-)modal inputs (e.g. `http://www.statmt.org/wmt16/multimodal-task.html`).

As a case study, we adapted two systems with non-textual inputs to run in OpenNMT. The first is an image-to-text system developed for mathematical OCR (Deng et al., 2016). This model replaces the source RNN with a deep convolution over the source input. Excepting preprocessing, the entire adaptation requires less than 500 lines of additional code and is also open-sourced as `github.com/opennmt/im2text`. The second is a speech-to-text recognition system based on the work of Chan et al. (2015). This system has been implemented directly in OpenNMT by replacing the source encoder with a Pyrimidal source model.

4.4 Additional Tools

Finally we briefly summarize some of the additional tools that extend OpenNMT to make it more beneficial to the research community.

Tokenization We aimed for OpenNMT to be a standalone project and not depend on commonly used tools. For instance the Moses tokenizer has language specific heuristics not necessary in NMT. We therefore include a simple reversible tokenizer that (a) includes markers seen by the model that allow simple deterministic detokenization, (b) has extremely simple, language-independent tokenization rules. The tokenizer can also perform Byte Pair Encoding (BPE) which has become a popular method for sub-word tokeniza-

	ES	FR	IT	PT	RO
ES	-	32.7 (+5.4)	28.0 (+4.6)	34.4 (+6.1)	28.7 (+6.4)
FR	32.9 (+3.3)	-	26.3 (+4.3)	30.9 (+5.2)	26.0 (+6.6)
IT	31.6 (+5.3)	31.0 (+5.8)	-	28.0 (+5.0)	24.3 (+5.9)
PT	35.3 (+10.4)	34.1 (+4.7)	28.1 (+5.6)	-	28.7 (+5.0)
RO	35.0 (+5.4)	31.9 (+9.0)	26.4 (+6.3)	31.6 (+7.3)	-

Table 2: 20 language pair single translation model. Table shows BLEU(Δ) where Δ compares to only using that pair for training.

Vocab	System	Speed tok/sec		BLEU
		Train	Trans	
V=50k	Nematus	3393	284	17.28
	ONMT	4185	380	17.60
V=32k	Nematus	3221	252	18.25
	ONMT	5254	457	19.34

Table 3: Performance Results for EN→DE on WMT15 tested on *newstest2014*. Both system 2x500 RNN, embedding size 300, 13 epochs, batch size 64, beam size 5. We compare on a 50k vocabulary and a 32k BPE setting.

tion in NMT systems (Sennrich et al., 2015).

Word Embeddings OpenNMT includes tools for simplifying the process of using pretrained word embeddings, even allowing automatic download of embeddings for many languages. This allows training in languages or domain with relatively little aligned data. Additionally OpenNMT can export the word embeddings from trained models to standard formats. This allows analysis is external tools such as TensorBoard, shown in Figure 4.4.

5 Benchmarks

We now document some runs of the model. We expect performance and memory usage to improve with further development. Public benchmarks are available at `http://opennmt.net/Models/`, which also includes publicly available pre-trained models for all of these tasks and tutorial instructions for all of these tasks. The benchmarks are run on a Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz, 256GB Mem, trained on 1 GPU GeForce GTX 1080 (Pascal) with CUDA v. 8.0 (driver 375.20) and cuDNN (v. 5005).

The comparison, shown in Table 3, is on English-to-German (EN→DE) using the WMT 2015[1] dataset. Here we compare, BLEU score, as well as training and test speed to the publicly available *Nematus* system. [2]

We additionally trained a multilingual translation model following Johnson (2016). The model translates from and to French, Spanish, Portuguese, Italian, and Romanian. Training data is 4M sentences and was selected from the open parallel corpus[3], specifically from Europarl, GlobalVoices and Ted. Corpus was selected to be multi-source, multi-target: each sentence has its translation in the 4 other languages. Corpus was tokenized using shared Byte Pair Encoding of 32k. Comparative results between multi-way translation and each of the 20 independent training are presented in Table 2. The systematically large improvement shows that language pair benefits from training jointly with the other language pairs.

Additionally we have found interest from the community in using OpenNMT for non-standard MT tasks like sentence document summarization dialogue response generation (chatbots), among others. Using OpenNMT, we were able to replicate the sentence summarization results of Chopra et al. (2016), reaching a ROUGE-1 score of 33.13 on the Gigaword data. We have also trained a model on 14 million sentences of the OpenSubtitles data set based on the work Vinyals and Le (2015), achieving comparable perplexity.

6 Conclusion

We introduce *OpenNMT*, a research toolkit for NMT that prioritizes efficiency and modularity. We hope to further develop OpenNMT to maintain strong MT results at the research frontier, providing a stable and framework for production use.

[1] `http://statmt.org/wmt15`

[2] `https://github.com/rsennrich/nematus`. Comparison with OpenNMT/Nematus github revisions `907824/75c6ab1`.

[3] `http://opus.lingfil.uu.se`

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation By Jointly Learning To Align and Translate. In *ICLR*. pages 1–15. https://doi.org/10.1146/annurev.neuro.26.041002.131047.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*. pages 10–21. http://aclweb.org/anthology/K/K16/K16-1002.pdf.

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2015. Listen, attend and spell. *CoRR* abs/1508.01211. http://arxiv.org/abs/1508.01211.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proc of EMNLP*.

Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. *Proceedings of NAACL-HLT16* pages 93–98.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Josep Crego, Jungi Kim, and Jean Senellart. 2016. Systran's pure neural machine translation system. *arXiv preprint arXiv:1602.06023* .

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. 2012. Large scale distributed deep networks. In *Advances in neural information processing systems*. pages 1223–1231.

Yuntian Deng, Anssi Kanervisto, and Alexander M. Rush. 2016. What you get is what you see: A visual markup decompiler. *CoRR* abs/1609.04938. http://arxiv.org/abs/1609.04938.

Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*. Association for Computational Linguistics, pages 7–12.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Mike Schuster Quoc V. Le Maxim Krikun Yonghui Wu Zhifeng Chen Nikhil Thorat Fernanda Vigas Martin Wattenberg Greg Corrado Macduff Hughes Jeffrey Dean Johnson. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation .

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc ACL*. Association for Computational Linguistics, pages 177–180.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proc of EMNLP*.

André FT Martins and Ramón Fernandez Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. *arXiv preprint arXiv:1602.02068* .

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. *arXiv preprint arXiv:1606.02892* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *CoRR* abs/1508.07909. http://arxiv.org/abs/1508.07909.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*. page 9. http://arxiv.org/abs/1409.3215.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869* .

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR* abs/1410.3916. http://arxiv.org/abs/1410.3916.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *CoRR* abs/1502.03044. http://arxiv.org/abs/1502.03044.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.