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.

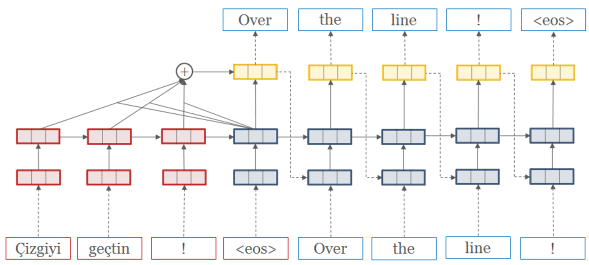

Figure 1: Schematic view of neural machine translation. The red source words are first mapped to word vectors and then fed into a recurrent neural network (RNN). Upon seeing the $\langle$eos$\rangle$ symbol, the final time step initializes a target blue RNN. At each target time step, *attention* is applied 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.

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 MT and the CDec toolkit (Dyer et al., 2010) for syntax-based MT, NMT toolkits can provide a foundation for the research community. 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.

Currently there are several existing NMT implementations. Many systems such as those developed in industry by Google (Wu et al., 2016), 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 this goal 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 report describes how the first-release of 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 preliminary 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 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 several other important aspects that contribute to model effectiveness: (a) It is important to use a gated RNN such as 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 layers (2-16) of RNN 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 learning, 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 system is implemented using the Torch mathematical framework and neural network library, and can be easily be extended using Torch's internal standard neural network components.

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 500 users, and there have been active development by those outside of these two organizations. The project has an active forum for community feedback.

One nice aspect of NMT as a model is its relative compactness. The complete OpenNMT system including preprocessing is roughly 4K lines of code. For comparison the Moses SMT framework including language modeling is over 100K lines. This makes the system easy to completely understand for newcomers and contributors. 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 in previous works, we focus this report on the design goals of OpenNMT. We focus particularly on three ordered criteria: 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.

Optimization: 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. This makes the system slightly less flexible than toolkits such as Element-RNN (Léonard et al., 2015), but provides a saving of 70% of GPU memory on the default model.

Optimization: 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).

Case Study: C/Mobile/GPU Translation Training NMT systems requires significant code complexity to facilitate fast back-propagation-through-time. At test time, 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. To exploit this asymmetry, OpenNMT includes several different decoders specialized for different run-time environments: a batched CPU/GPU decoder for very quickly translating a large set of sentences, a simple single-instance decoder for use on mobile devices, and a specialized C decoder. The last decoder is particularly suited for industrial use as it can run on CPU in standard production environments. The decoder reads the structure of the network from Lua and then uses the Eigen package to implement the basic linear algebra necessary for decoding.

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 a case features and model the probability of the lower-cased word form and the case marker. This extension requires modifying both the output of the decoder to generate multiple symbols, and also the input to the decoder to take in a word and its features. In OpenNMT both of these aspects are abstracted from the core translation code, and therefore we were able to add factored translation by modifying the input network to instead process the feature-based representation, and the output generator network to instead produce multiple conditionally independent predictions. This option can be turned on by modifying the training data to include the factored words.

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 times 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 NLP group developed a new method known as structured attention, that utilizes graphical model inference to compute this attention. The method was quite involved and required custom CUDA code to compute efficiently. However the method is modularized to fit the Torch NN interface and can therefore be directly used in OpenNMT to substitute for standard attention.

4.3 Extensibility

Deep learning is a very quick moving area, and we expect that there will soon be many other applications of NMT systems. Already we see related, but very different styles of work, such as variational seq2seq variation auto-encoders (Bowman et al., 2016) or memory networks (Weston et al., 2014). We developed a case study to ensure that OpenNMT was extensible to these changes.

Case Study: Image-to-Text In particular experimented with implementing a complete attention-based image-to-text translation system (Xu et al., 2015) using the OpenNMT library. This task is quick different than 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`). In particular, we adapted the im2markup system (Deng et al., 2016) to instead use OpenNMT as a library. This model replaces the source RNN with a deep convolution over the source input. However as this part of the network is pluggable, it could be naturally defined in Torch. In fact, excepting preprocessing, the entire adaptation requires only 500 lines of code and is also open-sourced as `http://github.com/opennmt/im2text`.

5 Benchmarks

In this section we document preliminary runs of the model. We expect performance and memory usage to improve with further development.

These benchmarks are run using a machine 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 (ver. 5005).

Vocabulary	Training Words/Sec	Translation Words/Sec	BLEU
Vocab 50k	4185/3393	380/284	17.6/17.28
BPE 32k	5254/3221	457/252	19.34/18.25

Table 1: Performance Results for EN→DE training on WMT15 corpus tested on newstest2014 - with 2 layers, RNN 500, wordvec size 500, 13 epochs, batch size 64. We compare *A*/*B* where *A* is training by OpenNMT, *B* is equivalent training with Nematus. OpenNMT/Nematus github revision `907824`/`75c6ab1`.

The first model is on English-to-German (EN→DE) using the WMT 2015[1] dataset. For comparison we also run the Nematus [2] system on the same data. Results are show in Table 5.

Additionally we also trained OpenNMT on several non-standard translation tasks. First is a summarization model () ...

Finally we trained a multilingual translation model following Johnson (2016). The model translates from and to French, Spanish, Portuguese, Italian, and Romanian (FR,ES,PT,IT,RO↔FR,ES,PT,IT,RO). Training data is 4M sentences and was selected from the open parallel corpus[3] and 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. The motivation of this selection was to evaluate the model cross-language learning. Corpus was tokenized using shared Byte Pair Encoding of 32k. Comparative results between multiway translation and each of the 20 independent training are presented in Table 5.

6 Conclusion and perspectives

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation By Jointly Learning To Align and Translate. In *ICLR*, pages 1–15.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous

[1] `http://statmt.org/wmt15`
[2] `https://github.com/rsennrich/nematus`
[3] `http://opus.lingfil.uu.se`

	ES	FR	IT	PT	RO
ES	-	32.71 (+5.43)	28 (+4.64)	34.41 (+6.08)	28.73 (+6.45)
FR	32.87 (+3.3)	-	26.32 (+4.25)	30.89 (+5.16)	25.95 (+6.64)
IT	31.64 (+5.34)	31.03 (+5.81)	-	27.96 (+4.98)	24.27 (+5.9)
PT	35.32 (+10.38)	34.08 (+4.68)	28.09 (+5.55)	-	28.73 (+5.03)
RO	35.00 (+5.37)	31.94 (+9.07)	26.38 (+6.34)	31.63 (+7.29)	-

Table 2: Performance Results for the twenty language pairs with the unique translation model. In parenthesis, score improvement compared to individual models trained with only the language pair data. The systematic huge improvement shows that each language pairs benefits from the learning of other language pairs which is remarkable since the corpus is completely parallel for the 5 languages which means that for a given language pair, no additional source or target sentence is presented to the system in another language pair.

space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 10–21.

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 *Proceedings of EMNLP*.

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.

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*, pages 7–12. Association for Computational Linguistics.

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

Quoc V. Le Maxim Krikun Yonghui Wu Zhifeng Chen Nikhil Thorat Fernanda Vigas Martin Wattenberg Greg Corrado Macduff Hughes Jeffrey Dean Johnson, Mike Schuster. 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 *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Nicholas Léonard, Sagar Waghmare, Yang Wang, and Jin-Hwa Kim. 2015. rnn : Recurrent library for torch. *CoRR*, abs/1511.07889.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*, page 9.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*, 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.

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*.