OpenNMT: Open-Source Toolkit for Neural Machine Translation

Yoon Kim*, **Guillaume Klein**†, **Yuntian Deng***, **Jean Senellart**†, **Alexander M. Rush***
Harvard University*, Systran †d

Abstract

We describe an open-source toolkit for neural machine translation that supports research development of sequence-to-sequence models. The toolkit prioritizes simplicity, modularity, and efficiency to make it reasonable for researchers to experiment with variants of neural machine translation that explore different feature representations, model architectures, and source (multi)-modalities, while maintaining competitive performance and tractable training requirements. The toolkit consists of modeling and decoding support, as well as detailed pedagogical documentation about the underlying methodologies.

1 Introduction

Neural machine translation (NMT) is a new methodology for machine translation (), that has shown remarkable gains particularly in terms of human evaluation. Originally popularized by the work of () and expanded upon by (). Over the last year the systems have been further refined and developed by work such as ().

In this work we describe a new open-source toolkit for developing neural machine translation systems, known as *OpenNMT*. The system is motivated by frameworks, such as Moses and CDec developed for statistical machine translation (SMT). These toolkits aim to provide a shared frameworks for developing and comparing open-source SMT systems that are complete and flexible enough for research development, while at the same time being efficient and accurate enough to be used production contexts.

Currently there are several existing NMT systems. Including Google NMT (), ... Several of these systems are proprietary. Several others are mainly research system such as seq2seq-attn. Perhaps most similar to OpenNMT is the NeMaTus system (), based on the system of () which aims to provide the same type of feature set as OpenNMT, using a different underlying neural network toolkit.

In developing OpenNMT we prioritized three different factors, which are described in this technical report: (1) Our top priority was training and decoding speed. NMT systems are notoriously slow to train, often requiring weeks of training time on the latest GPU hardware and significant test resources. We targeted this issue by implementing multi-GPU training, using aggressive memory sharing, and developing a specialized CPU decoder. (2) Our second priority was system modularity and teachibility. We intend OpenNMT as a living research system, and so the codebase was developed to provide a self-documenting overview of NMT. We discuss how this approach allowed us to add factored models () to the system. (3) Finally NMT is a very quick moving research area, and we would like the system to support new research areas as they develop. To demonstrate this approach we abstracted out the core of OpenNMT as a library, and describe the case study of using OpenNMT for image-to-text translation.

This report describes the background for NMT, the main aspects of system development, and the preliminary results from using the system in practice.

2 Background: Neural Machine Translation

Neural machine translation has now been extensively described in many excellent papers and tutorials (), which we briefly summarize. The arch

- One column describing the technical details

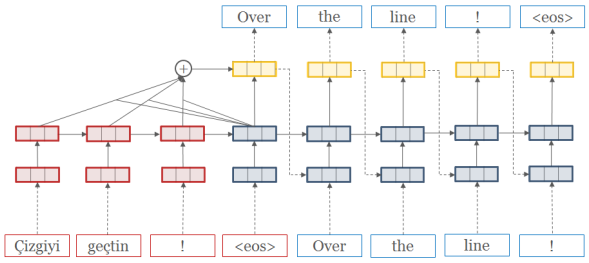

Figure 1: Schematic view of neural machine translation. The redred source words are first mapped to word vectors and then fed into a recurrent neural network (RNN). Upon seeing the ⟨eos⟩ symbol, the final time step initializes a target blueblue RNN. At each target time step, *attention* is applied over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

3 Implementation

Details: torch. text-to-text mapping. toolkit for extensibility

3.1 Modularity

- Separate encoder/decoder
- Arbitrary input/output representations (features, images)

3.2 Optimizations

Encoder/Decoder optimization

- Shared memory (cite opt net)
- C-Decoder

3.3 Advanced Features

Other papers .

4 User Studies

Feature-Based Inputs Discuss using modularity to add features as input and output

Knowledge Distillation/Pruning Discuss model access to allow for pruning and distillation.

Im2Latex Discuss ability to modify encoder representation to allow for other tasks.

5 Experimental Results

Mainly focus on NMT. Speed, memory, accuracy.
Picture of demo application running

Table 1: Performance Results. Several languages

Table 2: Speed Results. Multi-GPU, distillation, c decoder

6 Conclusion

References