F IRSEQ: Fast, Extensible Toolkit for Sequence Modeling

Myle Ott[△] **Sergey Edunov**[△] **lexei Baevski**[△] **ngela Fan**[△] **Sam Gross**[△]
Nathan Ng[△] **David Grangier**[▽†] **Michael uli**[△]
[△] Facebook I Research
[▽] Google Brain

bstract

F IRSEQ is an open-source sequence modeling toolkit that allows researchers and developers to train custom models for translation, summarization, language modeling, and other text generation tasks. The toolkit is based on PyTorch and supports distributed training across multiple GPUs and machines. We also support fast mixed-precision training and inference on modern GPUs. demo video can be found here: https://www.youtube.com/watch?v=OtgDdWtHvto.

1 Introduction

Neural sequence-to-sequence models have been successful on a variety of text generation tasks, including machine translation, abstractive document summarization, and language modeling. ccordingly, both researchers and industry professionals can benefit from a fast and easily extensible sequence modeling toolkit.

There are several toolkits with similar basic functionality, but they differ in focus area and intended audiences. For example, OpenNMT (Klein et al., 2017) is a community-built toolkit written in multiple languages with an emphasis on extensibility. MarianNMT (Junczys-Dowmunt et al., 2018) focuses on performance and the backend is written in C++ for fast automatic differentiation. OpenSeq2Seq (Kuchaiev et al., 2018) provides reference implementations for fast distributed and mixed precision training. Tensor2tensor (Vaswani et al., 2018) and Sockeye (Hieber et al., 2018) focus on production-readiness.

In this paper, we present F IRSEQ, a sequence modeling toolkit written in PyTorch that is fast, extensible, and useful for both research and production. F IRSEQ features: (i) a common interface across models and tasks that can be extended with user-supplied plug-ins (2); (ii) efficient distributed and mixed precision training, enabling training over datasets with hundreds of millions of sentences on current hardware (3); (iii) state-of-the-art implementations and pretrained models for machine translation, summarization, and language modeling (4); and (iv) optimized inference with multiple supported search algorithms, including beam search, diverse beam search (Vijayakumar et al., 2016), and top-k sampling. F IRSEQ is distributed with a BSD license and is available on GitHub at https://github.com/pytorch/fairseq.

2 Design

Extensibility. F IRSEQ can be extended through five types of user-supplied plug-ins, which enable experimenting with new ideas while reusing existing components as much as possible.

Models define the neural network architecture and encapsulate all learnable parameters. Models extend the `aseFairseqModel` class, which in turn extends `torch.nn.Module`. Thus any F IRSEQ model can be used as a stand-alone module in other PyTorch code. Models can additionally predefine named *architectures* with common network configurations (e.g., embedding dimension, number of layers, etc.). We also abstracted the methods through which the model interacts with the generation algorithm, e.g., beam search, through step-wise prediction. This isolates model implementation from the generation algorithm.

Criterions compute the loss given the model and a batch of data, roughly: `loss = criterion(model, batch)`. This formulation makes criterions very expressive, since they have complete access to the model. For example, a criterion may perform on-the-fly genera-

equal contribution
[†] Work done while at Facebook I Research.