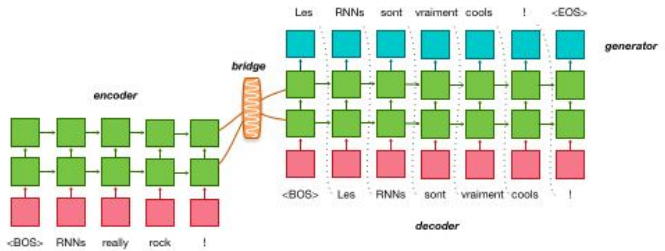
OpenNMT Tutorial

Presenter: Michael przystupa

Overview

- Background information
  - What is OpenNMT
  - Usage
  - Available Options out-of-the-box
- Training a default machine translation System
  - Translation tutorial
  - Stepping through preprocessing.py
  - Stepping through train.py
  - Stepping through translation.py

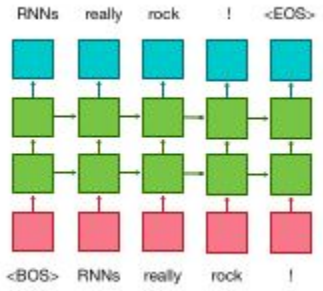


What is OpenNMT

- Open Source toolkit for Neural Sequence Models
- Target is production users
  - Designed to be moduler
- Most previous platforms are research code
  - GroundHog, Blocks, tensorflow-seq2seq, lamtram, seq2seq-attn
  - Nematus is exception to most of this
    - Well documented and lots of available options
    - OpenNMT largely inspired by them
- Built for 3 different deep learning APIs
  - Torch, PyTorch, TensorFlow

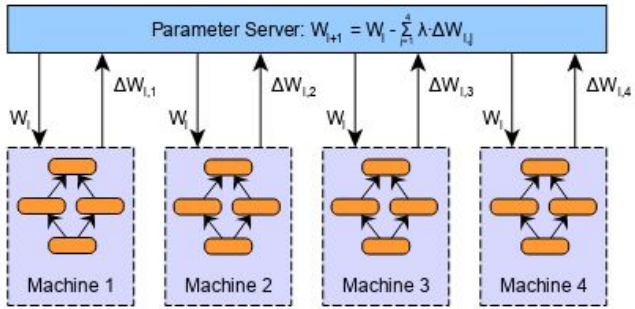
OpenNMT Usage

- Supports many scenarios (Tutorials for all):
  - Machine Translation
  - Summarization
  - Image to Text
  - Speech to Text
- Also can be used as a module
  - Quite brittle for pytorch at the moment



Available Options out-of-the-box

- Provides various options for end to end System:
  - Preprocessing data
    - E.g. building a vocabulary
  - Training model
    - Picking model, choosing hyperparameters, etc.
  - Testing Performance of Models
    - Generating predictions and calculate scores (e.g. BLEU)
- Supports Parallelism of Training
  - E.g. Multi-GPU support



Machine Translation in 3 commands

Preprocessing

```
python preprocess.py -train_src data/src-train.txt -train_tgt data/tgt-train.txt -valid_src data/src-val.txt -valid_tgt data/tgt-val.txt -save_data data/demo
```

Training

```
python train.py -data data/demo -save_model demo-model
```

Translate

```
python translate.py -model demo-model_XYZ.pt -src data/src-test.txt -output pred.txt -replace_unk -verbose
```

preprocessing.py

- My vague interpretation:
  a. Parse inputs
     - Check they are valid
  b. Create Preprocessing objects
     - Torchtext objects
  c. Read the datasets and save (in shards if necessary)
  d. Save your vocab

preprocessing.py

- <Open Pycharm>

train.py

- My vague interpretation:
  - Parse Configs, check asserts, determine if running in parallel
  - Create model: Encoder, decoder, load stuff (word embeddings or checkpoint model)
  - Build optimizer
  - Set-up trainer
  - Load dataset
  - Train Model

train.py

- <Open Pycharm>

translate.py

- My vague interpretation:
  - Handle parsing inputs
  - Create Translator
    - Load Saved model configs
    - Build the Translator class
  - Preprocess test data
  - Translate based on specifications

translate.py

- <Open Pycharm>