PYTORCH

VARIABLE-LENGTH MULTIMODAL INPUT & (IF TIME PERMITS) DATALOADER
HOW USEFUL IS IT?

• Machine Translation: two sequences of words with no predefined length
• ASR: variable-length audio and orthography
• Basically, anything you can do seq2seq upon
• Yes, I know `pytorch-seq2seq` is there. But what if you want something more hackable?
• Why don’t we just pad them to the same length?
• Padding leads to some other error-prone practices, including but not limited to:
  • keeping track of pre-padding length; slicing; wasting memory
WHAT WE’VE GOT

- PyTorch has CUDNN support, a CUDA library essential to speeding up RNN, among many others.
- With great performance comes strict formatting requirements. CUDNN dictates that every batch fed into accelerated RNN must be sorted by length of sequences.
- PyTorch solemnly swears that every RNN cell can take in a \texttt{torch.nn.utils.rnn.PackedSequence}, which is basically a batch of sorted sequences.
• from torch.utils.data import Dataset, DataLoader

• By implementing __len__(self) and __getitem__(self, index) methods, we can define a dataset. A dataloader takes in a dataset and specifies how batches should be formed.
SO WHAT’S WRONG?

• Say we have the following parallel text (English-Chinese)
• [“palatoglossus”, “back then when I left, willow trees wave me goodbye”]
• [”腭舌肌”,“昔我往矣杨柳依依”]
• Sorting them respectively will cause a mismatch.
HOW TO FIX IT

• Sort and remember how the batch is sorted (permutation indices), so that we can invert it later

• Workflow:
  • sort(sequences) → sorted_sequences + perm_index
  • pack_sequence(sorted_sequences) → PackedSequence
  • Feed to RNN
  • “unpack” the return PackedSequence with pad_packed_sequence
  • “unsort” the batch with inverted permutation index
LET’S LOOK AT A SMALL EXAMPLE
HOW DOES IT WORK WITH DATASET API?

• Instead of torch.Tensor, we may want to make dataset give us PackedSequence + perm_index

• It’s certainly easier said than done.
DATASET TRAPS

• When the Facebook folks market their flashing new framework and its dataset API, they claim that all you have to do is to override `__len__` and `__getitem__`.

• In fact, when dealing with variable-length data, it is a thing called `collate_fn` that usually matters.

How to create a dataloader with variable-size input

- **smth** • PyTorch Dev, Facebook AI Research

> you need to write a custom `collate_fn` and pass it to your data loader. [Questions about Dataloader and Dataset](https://discuss.pytorch.org/t/questions-about-dataloader-and-dataset/49314)
• A parameter of DataLoader that determines how batches are formed.
• It takes in a list of whatever type your `__getitem__` returns, and returns a “batch”.
• Really, I wish this parameter could be better named. Underscore, together with abbreviation makes it look unimportant.
LET’S LOOK AT AN EXAMPLE

• Data utils used in my recent research project