Exercise Briefing

Ki Hyun Kim

nlp.with.deep.learning@gmail.com

Objective:

- 샘플링 및 reward & loss 계산하기
  - Search 함수 활용

- 기존의 모델(e.g. Seq2seq, Transformer)들을 그대로 활용할 수 있어야 함
  - 기왕이면 기존 Trainer도 최대한 재활용 하는 방향으로

Review: REINFORCE with Baseline

Given policy $\pi_\theta(a|s)$ and value function v_ϕ(s),

For each episode:

Generate an episode $s_0, a_0, r_1, \cdots, s_{T-1}, a_{T-1}, r_T$ from π_θ.

Loop: update θ and ϕ for each step of the episode $t = 0, 1, \cdots, T-1$:

$$G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$$

$$\delta \leftarrow G - v_\phi(s_t)$$

$$\phi \leftarrow \phi + \eta^\phi \gamma^t \delta \nabla_\phi v_\phi(s_t)$$

$$\theta \leftarrow \theta + \eta^\theta \gamma^t \delta \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Equations

- What we will implement:

$$\mathcal{D} = \{x^i, y^i\}_{i=1}^N$$

$$\nabla_\theta \mathcal{L}(\theta) = \nabla_\theta \Big(\sum_{i=1}^{N} \log P(\hat{y}_0^i | x^i; \theta) \times -\big(\text{reward}(\hat{y}_0^i, y^i) - \frac{1}{K} \sum_{k=1}^{K} \text{reward}(\hat{y}_k^i, y^i)\big) \Big),$$

$$\text{where } \hat{y}^i \sim P(\cdot | x^i; \theta).$$

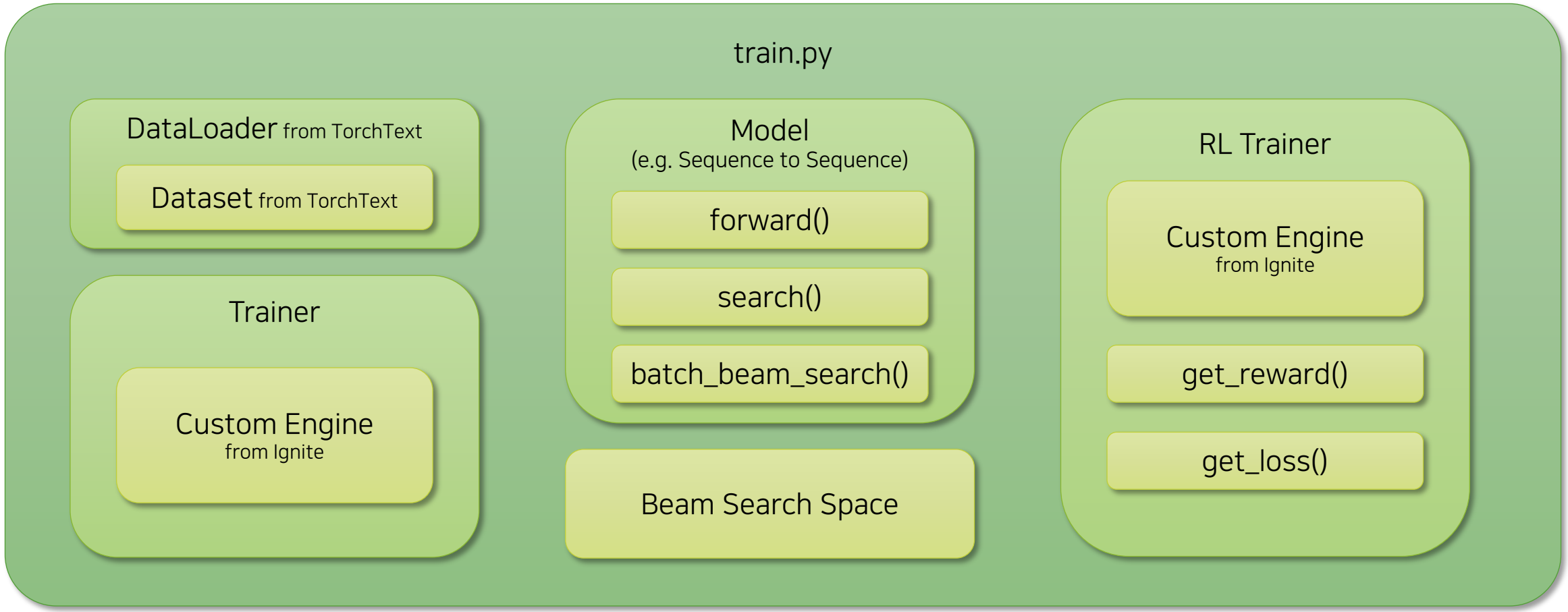
$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$$

Equations

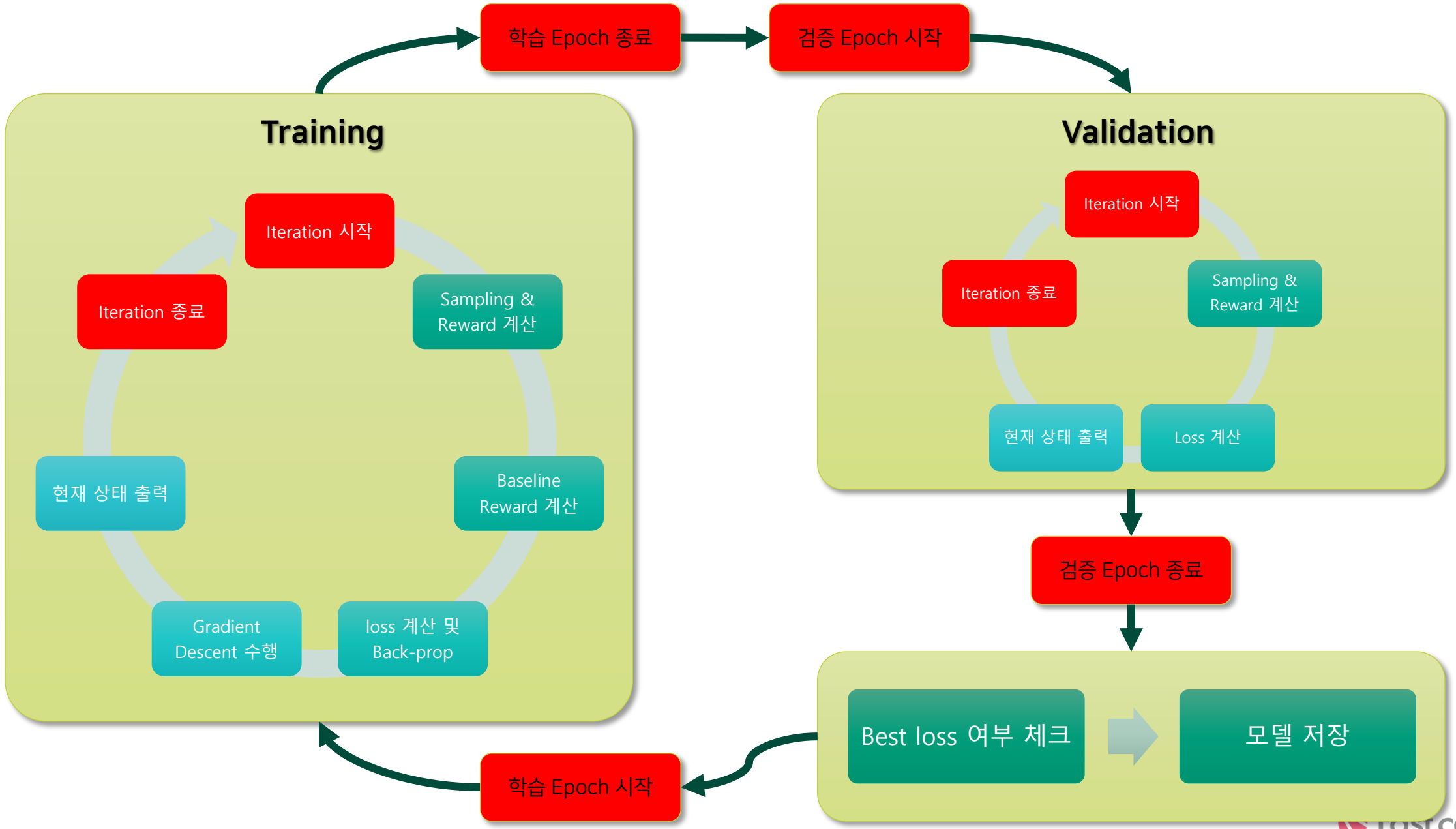
- How to implement:

$$\log P(\hat{y}|x;\theta) = \sum_{t=1}^{\tilde{m}} \hat{y}_t^{\mathsf{T}} \cdot \log f_\theta(x, \hat{y}_{<t})$$

Project Implementation



Training Procedure



Weird & Practical Tip

- Sort() is much faster? than Topk() in CUDA
  - https://github.com/pytorch/pytorch/issues/22812

```
import time
import torch

data = torch.rand(2000000, dtype=torch.float32, device=torch.device('cuda:0'))
num_topk = 1000

def topk1():
    return data.topk(num_topk, sorted=False)

def topk2():
    sort, idx = data.sort(descending=True)
    return sort[:num_topk], idx[:num_topk]

def benchmark(f, iter, warmup):
    for k in range(warmup): f()
    start = time.perf_counter()
    for k in range(iter): f()
    torch.cuda.synchronize()
    return time.perf_counter() - start

print(benchmark(topk1, 100, 3))
print(benchmark(topk2, 100, 3))
print(benchmark(topk1, 100, 3))
print(benchmark(topk2, 100, 3))
```

```
2.557645708322525
0.3197173401713371
2.5575470123440027
0.3197761407637596
```