Thuc hanh o nha_answer

December 20, 2021

1 Thực hành ở nhà Transformers

Hoàn thiện hàm huấn luyện cho mạng Transformer và tiến hành huấn luyện mô hình

1.0.1 Cài đặt giải thuật tối ưu và huấn luyện mô hình

[ ]:
```
import time

def train_model(model, opt):

    print("training model...")
    model.train()
    start = time.time()
    if opt.checkpoint > 0:
        cptime = time.time()

    for epoch in range(opt.epochs):

        total_loss = 0
        print("   %dm: epoch %d [%s]  %d%%  loss = %s" %\
            ((time.time() - start)//60, epoch + 1, "".join(' '*20), 0, '...'),␣
↪end='\r')

        if opt.checkpoint > 0:
            torch.save(model.state_dict(), 'weights/model_weights')

        for i, batch in enumerate(opt.train):

            src = batch.src.transpose(0,1).to(device)
            trg = batch.trg.transpose(0,1).to(device)
            trg_input = trg[:, :-1].to(device)
            src_mask, trg_mask = create_masks(src, trg_input, opt)
            preds = model(src, trg_input, src_mask, trg_mask)
            ys = trg[:, 1:].contiguous().view(-1)
            opt.optimizer.zero_grad()
            loss = F.cross_entropy(preds.view(-1, preds.size(-1)), ys,␣
↪ignore_index=opt.trg_pad)
            loss.backward()
```

```
            opt.optimizer.step()

            total_loss += loss.item()

            if (i + 1) % opt.printevery == 0:
                p = int(100 * (i + 1) / opt.train_len)
                avg_loss = total_loss/opt.printevery
                print("   %dm: epoch %d [%s%s]  %d%%  loss = %.3f" %\
                    ((time.time() - start)//60, epoch + 1, "".join('#'*(p//5)),␣
↪"".join(' '*(20-(p//5))), p, avg_loss))
                total_loss = 0

            if opt.checkpoint > 0 and ((time.time()-cptime)//60) // opt.
↪checkpoint >= 1:
                torch.save(model.state_dict(), 'weights/model_weights')
                cptime = time.time()


        print("%dm: epoch %d [%s%s]  %d%%  loss = %.3f\nepoch %d complete, loss␣
↪= %.03f" %\
        ((time.time() - start)//60, epoch + 1, "".join('#'*(100//5)), "".join('␣
↪'*(20-(100//5))), 100, avg_loss, epoch + 1, avg_loss))

class Opt(object):
    pass

def main():
    opt = Opt()
    opt.src_data = "data/english.txt"
    opt.trg_data = "data/french.txt"
    opt.src_lang = "en_core_web_sm"
    opt.trg_lang = 'fr_core_news_sm'
    opt.epochs = 2
    opt.d_model=512
    opt.n_layers=6
    opt.heads=8
    opt.dropout=0.1
    opt.batchsize=1500
    opt.printevery=100
    opt.lr=0.0001
    opt.max_strlen=80
    opt.checkpoint = 0
    opt.no_cuda = False
    opt.load_weights = None

    opt.device = 0
    if opt.device == 0:
```

```
        assert torch.cuda.is_available()

    read_data(opt)
    SRC, TRG = create_fields(opt)
    opt.train = create_dataset(opt, SRC, TRG)
    model = get_model(opt, len(SRC.vocab), len(TRG.vocab)).to(device)

    opt.optimizer = torch.optim.Adam(model.parameters(), lr=opt.lr, betas=(0.9,␣
 ↪0.98), eps=1e-9)

    if opt.checkpoint > 0:
        print("model weights will be saved every %d minutes and at end of epoch␣
 ↪to directory weights/"%(opt.checkpoint))

    train_model(model, opt)


    # for asking about further training use while true loop, and return
if __name__ == "__main__":
    main()
```