main.py文件中要补全的代码如下：

In [ ]:
```
class BertTagger(nn.Module):
    def __init__(self, hidden_dim, output_dim, model_name):
        super(BertTagger, self).__init__()
        # TODO:
        # （1）利用AutoConfig.from_pretrained定义config
        config = AutoConfig.from_pretrained(model_name)
        # （2）利用AutoModelWithLMHead.from_pretrained定义模型，注意要传入刚才的confi
        self.bert_model = AutoModelWithLMHead.from_pretrained(model_name, config=con
        # （3）定义一个线性层用于分类预测
        self.classifier = nn.Linear(config.hidden_size, output_dim)
        # 提示：参考文档https://huggingface.co/bert-base-cased

    def forward(self, X):
        # TODO:
        # （1）把X输入bert_model得到hidden_states;
        outputs = self.bert_model(X, attention_mask=torch.ones(X.shape).to(X.device)
        # （2）提取其中属于最后一个transformer layer的hidden_states作为最终特征;
        # （3）最后把特征输入线性层完成预测。
        # 提示：需要用到output_hidden_states参数，并参考以下文档
        # https://huggingface.co/docs/transformers/v4.21.2/en/model_doc/bert#transfo
        features =  outputs.hidden_states[-1]
        logits = logits = self.classifier(features)
        return logits
```

In [ ]:
```
def main(params):
    if params.seed:
        random.seed(params.seed)
        np.random.seed(params.seed)
        torch.manual_seed(params.seed)
        torch.cuda.manual_seed(params.seed)
        torch.backends.cudnn.deterministic = True
    logger = init_experiment(params, logger_filename=params.logger_filename)
    logger.info(params.__dict__)
    domain_name = os.path.basename(params.data_path[0])
    if domain_name == '':
        domain_name = os.path.basename(params.data_path[0][:-1])
    ner_dataloader = NER_dataloader(data_path=params.data_path,
                                    domain_name=domain_name,
                                    batch_size=params.batch_size,
                                    entity_list=params.entity_list)
    dataloader_train, dataloader_dev, dataloader_test = ner_dataloader.get_dataloade
    label_list = ner_dataloader.label_list
    entity_list = ner_dataloader.entity_list

    if params.model_name in ['bert-base-cased', 'roberta-base']:
        model = BertTagger(hidden_dim=params.hidden_dim,
                           output_dim=len(label_list),
                           model_name=params.model_name)
    else:
        raise Exception('model name %s is invalid' % params.model_name)
    model.cuda()
    trainer = BaseTrainer(params, model, entity_list, label_list)

    logger.info("Training ...")
    no_improvement_num = 0
    best_f1 = 0
    step = 0
    loss_history = []
    f1_history = []
```

```
logger.info("Initial lr is %s" % (str(trainer.scheduler.get_last_lr())))

for e in range(1, params.training_epochs+1):
    logger.info("============== epoch %d ==============" % e)
    loss_list = []
    mean_loss = 0.0
    total_cnt = 0
    correct_cnt = 0

    pbar = tqdm(dataloader_train, total=len(dataloader_train))
    for X, y in pbar:
        step += 1
        X, y = X.cuda(), y.cuda()
        trainer.batch_forward(X)
        correct_cnt += int(torch.sum(torch.eq(torch.argmax(trainer.logits, dim=2
        total_cnt += trainer.logits.size(0) * trainer.logits.size(1)
        trainer.batch_loss(y)
        loss = trainer.batch_backward()
        loss_list.append(loss)
        mean_loss = np.mean(loss_list)
        pbar.set_description("Epoch %d, Step %d: Loss=%.4f, Training_acc=%.2f%%"
            e, step, mean_loss, correct_cnt / total_cnt * 100
        ))
    loss_history.append(mean_loss)
    if params.info_per_epochs > 0 and e % params.info_per_epochs == 0:
        logger.info("Epoch %d, Step %d: Loss=%.4f, Training_acc=%.2f%%" % (
            e, step, mean_loss, correct_cnt / total_cnt * 100
        ))
    if trainer.scheduler != None:
        old_lr = trainer.scheduler.get_last_lr()
        trainer.scheduler.step()
        new_lr = trainer.scheduler.get_last_lr()
        if old_lr != new_lr:
            logger.info("Epoch %d, Step %d: lr is %s" % (
                e, step, str(new_lr)
            ))
    if params.save_per_epochs != 0 and e % params.save_per_epochs == 0:
        trainer.save_model("best_finetune_domain_%s_epoch_%d.pth" % (domain_name
    if e % params.evaluate_interval == 0:
        f1_dev, f1_dev_each_class = trainer.evaluate(dataloader_dev, each_class=
        logger.info("Epoch %d, Step %d: Dev_f1=%.4f, Dev_f1_each_class=%s" % (
            e, step, f1_dev, str(f1_dev_each_class)
        ))
        f1_history.append(f1_dev)
        if f1_dev > best_f1:
            logger.info("Find better model!!")
            best_f1 = f1_dev
            no_improvement_num = 0
            trainer.save_model("best_finetune_domain_%s.pth" % domain_name, path
        else:
            no_improvement_num += 1
            logger.info("No better model is found (%d/%d)" % (no_improvement_num
        if no_improvement_num >= params.early_stop:
            logger.info("Stop training because no better model is found!!!")
            break
logger.info("Finish training ...")

logger.info("Testing...")
trainer.load_model("best_finetune_domain_%s.pth" % domain_name, path=params.dump
trainer.model.cuda()
f1_test, f1_score_dict = trainer.evaluate(dataloader_test, each_class=True)
logger.info("Final Result: Evaluate on Test Set. F1: %.4f." % (f1_test))
f1_score_dict = sorted(f1_score_dict.items(), key=lambda x: x[0])
```

```
    logger.info("F1_list: %s" % (f1_score_dict))
    logger.info("Finish testing ...")

    # Visualize the training process
    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.plot(loss_history, label='Training Loss')
    plt.title('Training Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(f1_history, label='Dev F1 Score')
    plt.title('Dev F1 Score')
    plt.xlabel('Epoch')
    plt.ylabel('F1 Score')
    plt.legend()

    plt.tight_layout()
    plt.savefig(os.path.join(params.dump_path, 'training_process.png'))
    plt.show()

if __name__ == "__main__":
    params = get_params()
    main(params)
```

运行程序后，得到的训练过程以及可视化结果图片如下：

Some weights of BertForTokenClassification were not initialized from the model checkpoint at bert-base-cased and are newly initialized: ['classifier.bias', 'classifier.weight'] You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference. INFO - 11/11/24 09:31:04 - 0:01:04 - Training ... INFO - 11/11/24 09:31:04 - 0:01:04 - Initial lr is [0.001] INFO - 11/11/24 09:31:04 - 0:01:04 - ============== epoch 1 ============== Epoch 1, Step 878: Loss=0.2574, Training_acc=32.33%: 100%|

878/878 [01:50<00:00, 7.98it/s] INFO - 11/11/24 09:32:54 - 0:02:54 - Epoch 1, Step 878: Loss=0.2574, Training_acc=32.33% INFO - 11/11/24 09:33:04 - 0:03:04 - Epoch 1, Step 878: Dev_f1=85.2552, Dev_f1_each_class={'location': 88.1, 'misc': 67.62, 'organisation': 79.36, 'person': 95.33} INFO - 11/11/24 09:33:04 - 0:03:04 - Find better model!! INFO - 11/11/24 09:33:04 - 0:03:05 - Best model has been saved to experiments/default/1/best_finetune_domain_conll2003.pth INFO - 11/11/24 09:33:04 - 0:03:05 - ============== epoch 2 ============== Epoch 2, Step 1756: Loss=0.0891, Training_acc=33.76%: 100%|

878/878 [01:53<00:00, 7.73it/s] INFO - 11/11/24 09:34:58 - 0:04:59 - Epoch 2, Step 1756: Loss=0.0891, Training_acc=33.76% INFO - 11/11/24 09:35:08 - 0:05:08 - Epoch 2, Step 1756: Dev_f1=89.7029, Dev_f1_each_class={'location': 91.77, 'misc': 77.27, 'organisation': 85.85, 'person': 96.58} INFO - 11/11/24 09:35:08 - 0:05:08 - Find better model!! INFO - 11/11/24 09:35:11 - 0:05:11 - Best model has been saved to experiments/default/1/best_finetune_domain_conll2003.pth INFO - 11/11/24 09:35:11 - 0:05:11 - ============== epoch 3 ============== Epoch 3, Step 2634: Loss=0.0609, Training_acc=34.00%: 100%|


878/878 [01:54<00:00, 7.65it/s] INFO - 11/11/24 09:37:05 - 0:07:06 - Epoch 3, Step 2634: Loss=0.0609, Training_acc=34.00% INFO - 11/11/24 09:37:05 - 0:07:06 - Epoch 3, Step 2634: lr is [0.0002] INFO - 11/11/24 09:37:15 - 0:07:16 - Epoch 3, Step 2634: Dev_f1=91.2581, Dev_f1_each_class={'location': 93.4, 'misc': 80.63, 'organisation': 88.77, 'person': 96.34} INFO - 11/11/24 09:37:15 - 0:07:16 - Find better model!! INFO - 11/11/24 09:37:18 - 0:07:18 - Best model has been saved to experiments/default/1/best_finetune_domain_conll2003.pth INFO - 11/11/24 09:37:18 - 0:07:18 - ========== epoch 4 ========== Epoch 4, Step 3512: Loss=0.0439, Training_acc=34.15%: 100%|

878/878 [01:54<00:00, 7.64it/s] INFO - 11/11/24 09:39:13 - 0:09:13 - Epoch 4, Step 3512: Loss=0.0439, Training_acc=34.15% INFO - 11/11/24 09:39:23 - 0:09:23 - Epoch 4, Step 3512: Dev_f1=92.0887, Dev_f1_each_class={'location': 94.33, 'misc': 82.05, 'organisation': 89.75, 'person': 96.54} INFO - 11/11/24 09:39:23 - 0:09:23 - Find better model!! INFO - 11/11/24 09:39:25 - 0:09:26 - Best model has been saved to experiments/default/1/best_finetune_domain_conll2003.pth INFO - 11/11/24 09:39:25 - 0:09:26 - ========== epoch 5 ========== Loss=0.0410, Training_acc=34.17%: 100%|

Epoch 5, Step 4390: Loss=0.0410, Training_acc=34.17%: 100%|

| 878/878 [1:06:03<00:00, 4.51s/it] INFO - 11/11/24 10:45:29 - 1:15:29 - Epoch 5, Step 4390: Loss=0.0410, Training_acc=34.17% INFO - 11/11/24 10:45:38 - 1:15:39 - Epoch 5, Step 4390: Dev_f1=92.1352, Dev_f1_each_class={'location': 94.11, 'misc': 82.66, 'organisation': 90.34, 'person': 96.2} INFO - 11/11/24 10:45:38 - 1:15:39 - Find better model!! INFO - 11/11/24 10:45:41 - 1:15:41 - Best model has been saved to experiments/default/1/best_finetune_domain_conll2003.pth INFO - 11/11/24 10:45:41 - 1:15:41 - ========== epoch 6 ========== Epoch 6, Step 5268: Loss=0.0380, Training_acc=34.20%: 100%|

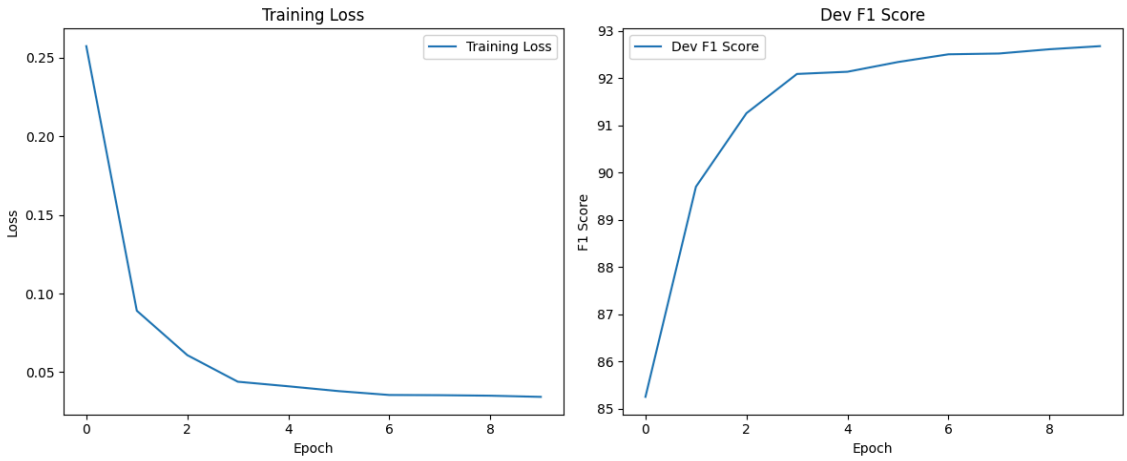
| 878/878 [01:51<00:00, 7.85it/s] INFO - 11/11/24 10:47:32 - 1:17:33 - Epoch 6, Step 5268: Loss=0.0380, Training_acc=34.20% INFO - 11/11/24 10:47:32 - 1:17:33 - Epoch 6, Step 5268: lr is [4e-05] INFO - 11/11/24 10:47:42 - 1:17:43 - Epoch 6, Step 5268: Dev_f1=92.3412, Dev_f1_each_class={'location': 94.37, 'misc': 82.87, 'organisation': 90.59, 'person': 96.34} INFO - 11/11/24 10:47:42 - 1:17:43 - Find better model!! INFO - 11/11/24 10:47:45 - 1:17:45 - Best model has been saved to experiments/default/1/best_finetune_domain_conll2003.pth INFO - 11/11/24 10:47:45 - 1:17:45 - ========== epoch 7 ========== Epoch 7, Step 6146: Loss=0.0355, Training_acc=34.21%: 100%|

| 878/878 [01:54<00:00, 7.70it/s] INFO - 11/11/24 10:49:39 - 1:19:39 - Epoch 7, Step 6146: Loss=0.0355, Training_acc=34.21% INFO - 11/11/24 10:49:49 - 1:19:49 - Epoch 7, Step 6146: Dev_f1=92.5046, Dev_f1_each_class={'location': 94.46, 'misc': 83.15, 'organisation': 90.8, 'person': 96.47} INFO - 11/11/24 10:49:49 - 1:19:49 - Find better model!! INFO - 11/11/24 10:49:51 - 1:19:52 - Best model has been saved to experiments/default/1/best_finetune_domain_conll2003.pth INFO - 11/11/24 10:49:51 - 1:19:52 - ========== epoch 8 ========== Epoch 8, Step 7024: Loss=0.0354, Training_acc=34.22%: 100%|

| 878/878 [01:54<00:00, 7.67it/s] INFO - 11/11/24 10:51:46 - 1:21:46 - Epoch 8, Step 7024: Loss=0.0354, Training_acc=34.22% INFO - 11/11/24 10:51:56 - 1:21:56 - Epoch 8, Step 7024: Dev_f1=92.5213, Dev_f1_each_class={'location': 94.51, 'misc': 83.51, 'organisation': 90.61, 'person': 96.44} INFO - 11/11/24

10:51:56 - 1:21:56 - Find better model!! INFO - 11/11/24 10:51:58 - 1:21:59 - Best model has been saved to experiments/default/1/best_finetune_domain_conll2003.pth INFO - 11/11/24 10:51:58 - 1:21:59 - ============== epoch 9 ============== Epoch 9, Step 7902: Loss=0.0351, Training_acc=34.22%: 100%|████| 878/878 [01:54<00:00, 7.66it/s] INFO - 11/11/24 10:53:53 - 1:23:53 - Epoch 9, Step 7902: Loss=0.0351, Training_acc=34.22% INFO - 11/11/24 10:54:03 - 1:24:03 - Epoch 9, Step 7902: Dev_f1=92.6115, Dev_f1_each_class={'location': 94.57, 'misc': 83.93, 'organisation': 90.73, 'person': 96.39} INFO - 11/11/24 10:54:03 - 1:24:03 - Find better model!! INFO - 11/11/24 10:54:06 - 1:24:06 - Best model has been saved to experiments/default/1/best_finetune_domain_conll2003.pth INFO - 11/11/24 10:54:06 - 1:24:06 - ============== epoch 10 ============== Epoch 10, Step 8780: Loss=0.0343, Training_acc=34.22%: 100%|████| 878/878 [01:54<00:00, 7.66it/s] INFO - 11/11/24 10:56:00 - 1:26:01 - Epoch 10, Step 8780: Loss=0.0343, Training_acc=34.22% INFO - 11/11/24 10:56:10 - 1:26:11 - Epoch 10, Step 8780: Dev_f1=92.6772, Dev_f1_each_class={'location': 94.65, 'misc': 84.08, 'organisation': 90.75, 'person': 96.42} INFO - 11/11/24 10:56:10 - 1:26:11 - Find better model!! INFO - 11/11/24 10:56:13 - 1:26:13 - Best model has been saved to experiments/default/1/best_finetune_domain_conll2003.pth INFO - 11/11/24 10:56:13 - 1:26:13 - Finish training ... INFO - 11/11/24 10:56:13 - 1:26:13 - Testing... INFO - 11/11/24 10:56:13 - 1:26:14 - Model has been load from experiments/default/1/best_finetune_domain_conll2003.pth INFO - 11/11/24 10:56:22 - 1:26:23 - Final Result: Evaluate on Test Set. F1: 89.7995. INFO - 11/11/24 10:56:22 - 1:26:23 - F1_list: [('location', 92.06), ('misc', 74.59), ('organisation', 88.34), ('person', 95.82)] INFO - 11/11/24 10:56:22 - 1:26:23 - Finish testing ...

可视化结果如下，对loss和f1随epoch进行了可视化：



训练结束后,利用predict.py载入保存的模型，并输入自定义的例子进行预测，分析模型的输出结果；

输入： Japan began the defence of their Asian Cup title with a lucky 2 - 1 win against Syria in a Group C championship match on Friday 输出： ['B-location' 'O' 'O' 'O' 'O' 'O' 'B-misc' 'I-misc' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'B-location' 'O' 'O' 'B-misc' 'I-misc' 'O' 'O' 'O' 'O']

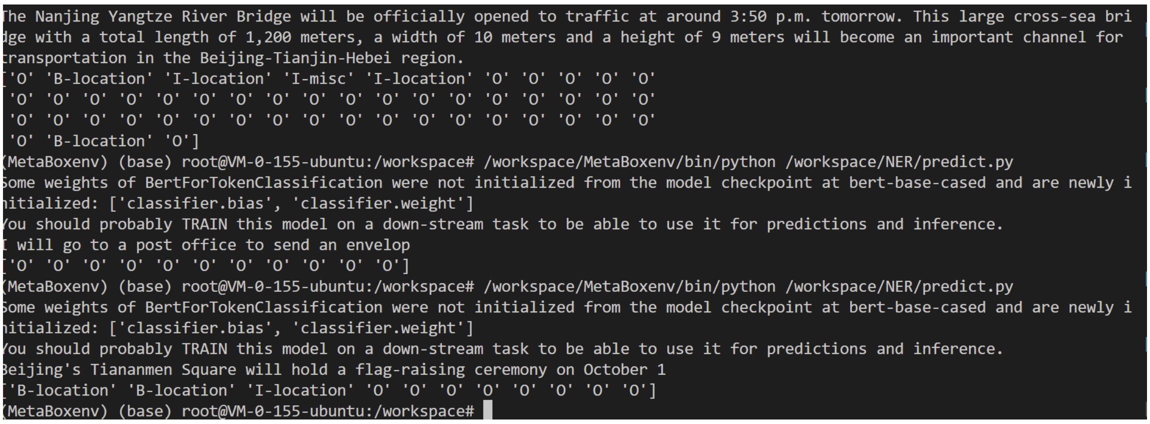
输入: I will go to a post office to send an envelop 输出: ['O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O']

输入: The Nanjing Yangtze River Bridge will be officially opened to traffic at around 3:50 p.m. tomorrow. This large cross-sea bridge with a total length of 1,200 meters, a width of 10 meters and a height of 9 meters will become an important channel for transportation in the Beijing-Tianjin-Hebei region. 输出: ['O' 'B-location' 'I-location' 'I-misc' 'I-location' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'B-location' 'O']

输入: Beijing's Tiananmen Square will hold a flag-raising ceremony on October 1 输出: ['B-location' 'B-location' 'I-location' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O']

输入: Huang Wei will visit my home 输出: ['B-person' 'I-person' 'O' 'O' 'O' 'O']

输入: The World Health Organization announced that it would step up epidemic prevention measures 输出: ['O' 'B-organisation' 'I-organisation' 'I-organisation' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O' 'O']



回答思考题：① CoNLL2003有4种实体类型，为什么输出的维度是9；②为什么需要额外加线性层用于预测，而不用transformers模型原有自带的线性层；③在predict.py当中，tokenizer的作用是什么；④在predict.py当中，mask的作用是什么。

1.任务通常需要区分实体的开始和内部。因此，对于每种实体类型，有“B-”（实体开始）和“I-”（实体内部）两种形式，4种实体类型就有8种标签。再加上“O”（非实体）标签，总共就有9种标签。

2.原有的线性层是用于语言模型预测任务的，它的输出维度通常与词汇表大小相匹配。而在NER任务中，我们需要的输出是实体标签，其数量远小于词汇表大小。所以需要一个额外的线性层来将BERT的隐藏状态映射到正确的标签数量上。

3.将文本字符串转换为模型可以理解的格式。它将文本分解成单词或子词单元（tokens），并将这些tokens转换为模型的输入ID。

4.用于指示模型哪些部分的输入是实际数据，哪些部分是填充数据。在处理批次数据时，不同序列的长度可能不同，因此需要用填充来确保所有序列长度一致。Mask帮助模型忽略这些填充部分，只关注实际的数据。

说明现有模型不足之处并改进模型，展示改进模型的性能。比如，增加CRF层。

从我运行预测结果来看，训练的模型对于一些实体识别的不精确比如： Beijing's Tiananmen Square 应该是一个实体但是却识别为了两个实体，说明模型对于序列依赖性处理的不好，可以通过添加CRF层更好地捕捉标签之间的依赖关系，去理解实体的开始和结束。

下面是做的代码上的一些修改：

In [ ]:
```
import torch
import torch.nn as nn
from transformers import AutoConfig, AutoModelWithLMHead
from torchcrf import CRF

class BertTagger(nn.Module):
    def __init__(self, hidden_dim, output_dim, model_name):
        super(BertTagger, self).__init__()
        config = AutoConfig.from_pretrained(model_name)
        self.bert_model = AutoModelWithLMHead.from_pretrained(model_name, config=con
        self.classifier = nn.Linear(config.hidden_size, output_dim)
        self.crf = CRF(num_tags=output_dim, batch_first=True)

    def forward(self, X, mask):
        outputs = self.bert_model(X, attention_mask=mask, output_hidden_states=True
        hidden_states = outputs.hidden_states[-1]
        logits = self.classifier(hidden_states)
        return logits

    def loss(self, X, y, mask):
        logits = self.forward(X, mask)
        loss = self.crf(emissions=logits, tags=y, mask=mask)
        return loss
```

In [ ]:
```
class BaseTrainer(object):
    def __init__(self, params, model, entity_list, label_list):
        # parameters
        self.params = params
        self.model = model
        self.label_list = label_list
        self.entity_list = entity_list

        # training
        self.lr = float(params.lr)
        self.early_stop = params.early_stop
        self.no_improvement_num = 0

        self.mu = 0.9
        self.weight_decay = 5e-4

        # build scheduler and optimizer
        self.optimizer = torch.optim.SGD(self.model.parameters(),
                                         lr=self.lr,
                                         momentum=self.mu,
                                         weight_decay=self.weight_decay)
        self.scheduler = torch.optim.lr_scheduler.MultiStepLR(self.optimizer,mileste

    def batch_forward(self, inputs):
        self.logits = self.model.forward(inputs[0], inputs[1])

    def batch_loss(self, labels):
        self.loss = self.model.loss(self.logits, labels, inputs[1])

    def batch_backward(self):
        self.model.train()
        self.optimizer.zero_grad()
        self.loss.backward()
        self.optimizer.step()

        return self.loss.item()
```

```
    def evaluate(self, dataloader, each_class=False, entity_order=[]):
        with torch.no_grad():
            self.model.eval()
            y_list = []
            x_list = []
            logits_list = []
            mask_list = []

            for x, y in dataloader:
                x, y = x.cuda(), y.cuda()
                self.batch_forward((x, x))
                _logits = self.logits.view(-1, self.logits.shape[-1]).detach().cpu(
                logits_list.append(_logits)
                x = x.view(x.size(0)*x.size(1)).detach().cpu()
                x_list.append(x)
                y = y.view(y.size(0)*y.size(1)).detach().cpu()
                y_list.append(y)
                mask = (x != self.params.pad_token_label_id).float().cuda()
                mask_list.append(mask)

            y_list = torch.cat(y_list)
            x_list = torch.cat(x_list)
            logits_list = torch.cat(logits_list)
            pred_list = self.model.crf.decode(logits_list, mask_list[0])

            pred_line = []
            gold_line = []
            for pred_index, gold_index in zip(pred_list, y_list):
                gold_index = int(gold_index)
                if gold_index != self.params.pad_token_label_id:
                    pred_token = self.label_list[pred_index]
                    gold_token = self.label_list[gold_index]
                    pred_line.append(pred_token)
                    gold_line.append(gold_token)

            f1 = f1_score([gold_line], [pred_line])*100
            if not each_class:
                return f1

            f1_list = f1_score([gold_line], [pred_line], average=None)
            f1_list = list(np.array(f1_list)*100)
            gold_entity_set = set()
            for l in np.unique(gold_line):
                if 'B-' in l or 'I-' in l:
                    gold_entity_set.add(l[2:])
            gold_entity_list = sorted(list(gold_entity_set))
            f1_score_dict = dict()
            for e, s in zip(gold_entity_list, f1_list):
                f1_score_dict[e] = round(s, 2)
            if entity_order == []:
                return f1, f1_score_dict
            assert set(entity_order) == set(gold_entity_list), "gold_entity_list an
            ordered_f1_score_dict = dict()
            for e in entity_order:
                ordered_f1_score_dict[e] = f1_score_dict[e]
            return f1, ordered_f1_score_dict
```