NJU_NLP_SummerCamp_2019_report_week2

> 2019.07.08~2019.07.14
> @author Eric ZHU

本周学习内容

1. RNN，LSTM等对于序列数据的处理方法
2. 基于BiLSTM与CRF的NER任务模型 （基于Tensorflow与Keras）
3. 课程内容：基础的强化学习（Q-learning与DQN）

其他说明

1. 因为这两天在赶院内课程作业的DDL（基于DQN的Atari游戏AI），必须以Tensorflow作为框架，所以本周暂时继续使用Tensorflow；做完本次作业后就换到Pytorch（下学期院里的机器学习课程应该也是用Pytorch，也是需要提前换过来了解的）
2. 还是因为这两天在赶DDL，RL模型需要的训练与调试时间比较长，所以这周的实验报告没能抽出足够时间写，三个实验（LSTM，CRF+BiLSTM，调参）合并写了一次实验报告，见谅。此外，考虑到后面的任务可能也会用到RL，我把作业目前的进度也一并附在了此次的周报里，可供参考。

附录

1. 7.8-7.14_NER 实验记录（见下页）
2. 7.8-7.14_DQN 实验记录（见下页）

基于BiLSTM+CRF的中文NER任务

> 时间：2019-07-11
> 以下部分代码参考了这篇文章
> 此处为本文GitHub代码库链接

文件结构

名称	描述
data\	所用的数据集（来源：链接）
model\	保存的训练完成的模型文件
img\	文档中展示的部分图片
NER_data.py	数据预处理脚本
NER_model.py	模型的构造脚本
NER_train.py	模型的训练脚本
NER_eval.py	模型表现的评估脚本

测试环境

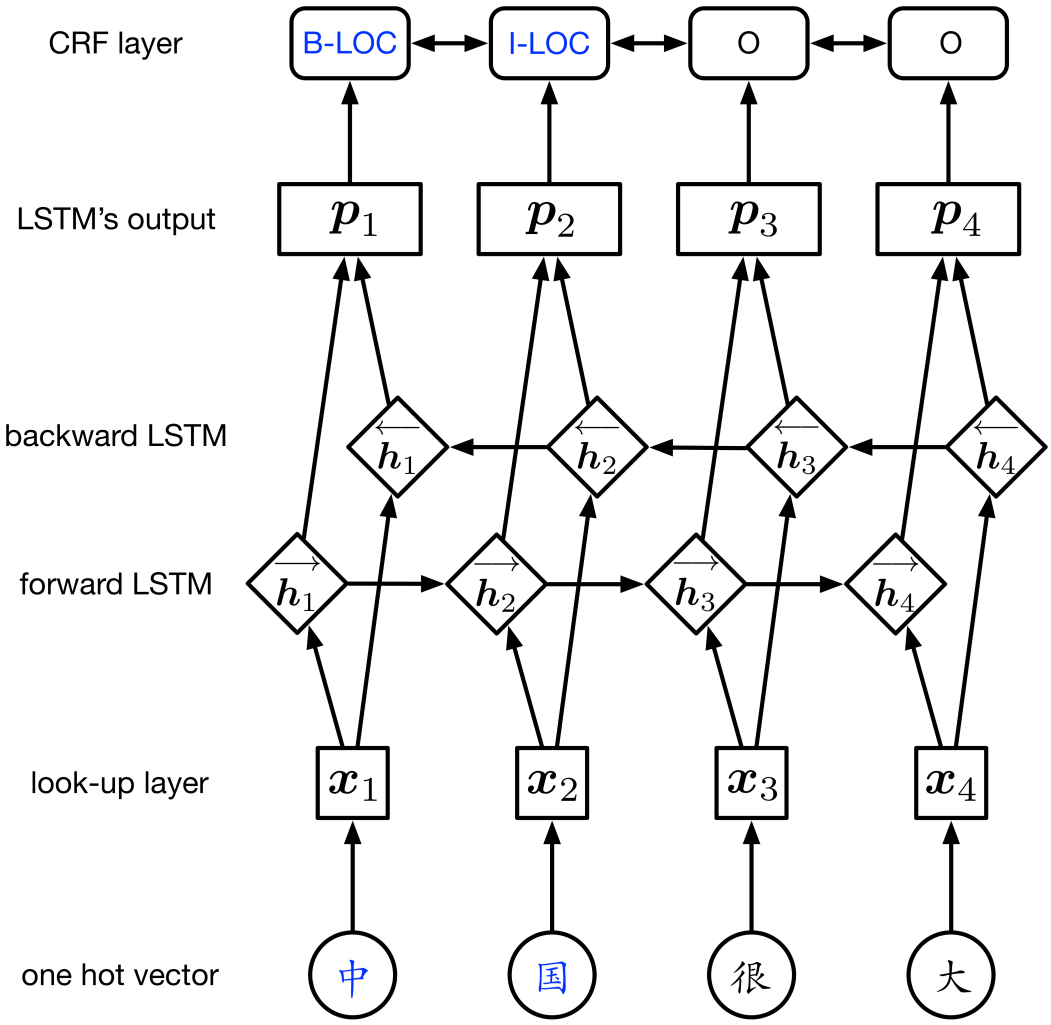
> 使用的测试环境如下：
>
> Tensorflow-gpu = 1.14.0
> Keras = 2.2.4
> Numpy = 1.16.4
>
> 训练机配置如下：
>
> CPU : I7-8700 (6C12T)
> GPU: RTX2060 (6G)
> CUDA = v10.0
> cuDNN = v7.3.1

处理流程

1. 数据预处理：读取训练数据，构造词典库。
2. 模型构造：初始化网络模型。
3. 模型训练：用训练集（共20864个句子）进行训练
4. 模型评估：用测试集（共4636个句子）对模型的Precision 、Recall与F1-score进行评估

模型结构

模型结构参考了此项目的模型结构，具体结构如下（图片来自此链接）



1. look-up layer

将每个字符的表示形式由 One-hot 向量转化为 Character Embedding. （此处的Embedding为随机初始化得到，效果较差，可考虑采用训练完成的Embedding进行初始化，可较大幅度地提升模型的表现）

2. BiLSTM layer

双向的LSTM层，将句子与词语中的特征进行抽取。BiLSTM相较单向的LSTM而言可以更好地捕捉双向的语义依赖。

3. CRF layer

为句子中的每个字符生成标签。CRF相较于Softmax可更好的利用字符在语句层面上包含的信息，从而提高分类的准确性。

模型调整

Batch_size = 16, Epoches = 3

类别	Precision	Recall	F1-score
O	98.36%	98.91%	0.9863
B-PER	61.76%	70.00%	0.6562
I-PER	59.65%	73.91%	0.6602
B-LOC	76.00%	71.25%	0.7355
I-LOC	68.03%	66.94%	0.6748
B-ORG	67.14%	74.60%	0.7068
I-ORG	77.94%	49.53%	0.6057
Mean	96.97%	97.08%	0.9697

Batch_size = 32, Epoches = 4

类别	Precision	Recall	F1-score
O	98.58%	98.68%	0.9863
B-PER	57.14%	70.59%	0.6316
I-PER	53.12%	68.00%	0.5965
B-LOC	68.75%	57.89%	0.6286
I-LOC	69.23%	49.09%	0.5745
B-ORG	47.62%	47.62%	0.4762
I-ORG	67.14%	74.60%	0.7068
Mean	96.88%	96.87%	0.9684

问题反思

1. 数据量较小，仅有约2万个句子作为训练集，导致模型表现不佳。
2. 采用了宽度为100*2的BiLSTM进行特征抽取，为此会将所有句子统一padding为长度200后进行输入。因此，调整LSTM层的宽度没有对模型的表现产生明显的影响。

基于Dueling DQN的Atari游戏AI模型

> 时间：2019-07-14
> 以下部分代码参考了 UCB CS294 HW3 中的课程代码，代码库链接

项目文件结构

名称	描述
log\	实验过程中输出的日志文件
experiments\	训练过程中保存的模型
img\	文档中展示的部分图片
atari_wrappers.py	调用OpenAI gym Atari 环境接口的脚本
logz.py	训练过程的记录脚本
DQN_utils.py	DQN依赖脚本，定义了损失函数，学习率变化情况等信息
DQN_learn.py	DQN学习脚本，定义了具体的学习过程
DQN_train.py	DQN训练脚本，模型的训练入口

测试环境

> 使用的测试环境如下：
>
> Python == 3.6
> tensorflow == 1.14.0 (GPU version)
> gym == 0.10.5 (注意最新版本gym可能无法正常运行)
> opencv-python == 4.1.0.25
> numpy == 1.16.4
>
> 训练机配置如下：
>
> CPU : I7-8700 (6C12T)
> GPU: RTX2060 (6G)
> CUDA = v10.0
> cuDNN = v7.3.1

处理流程

1. 初始化gym环境，取连续4帧图像，进行二值化，裁剪后作为输入数据。
2. 初始化网络模型，填充Replay Buffer
3. 对Replay Buffer进行采样，基于Exploration rate作出决策，用反馈的reward更新q_funtion（调整网络模型的参数）
4. 持续训练，对训练过程中的数据进行输出

模型网络结构

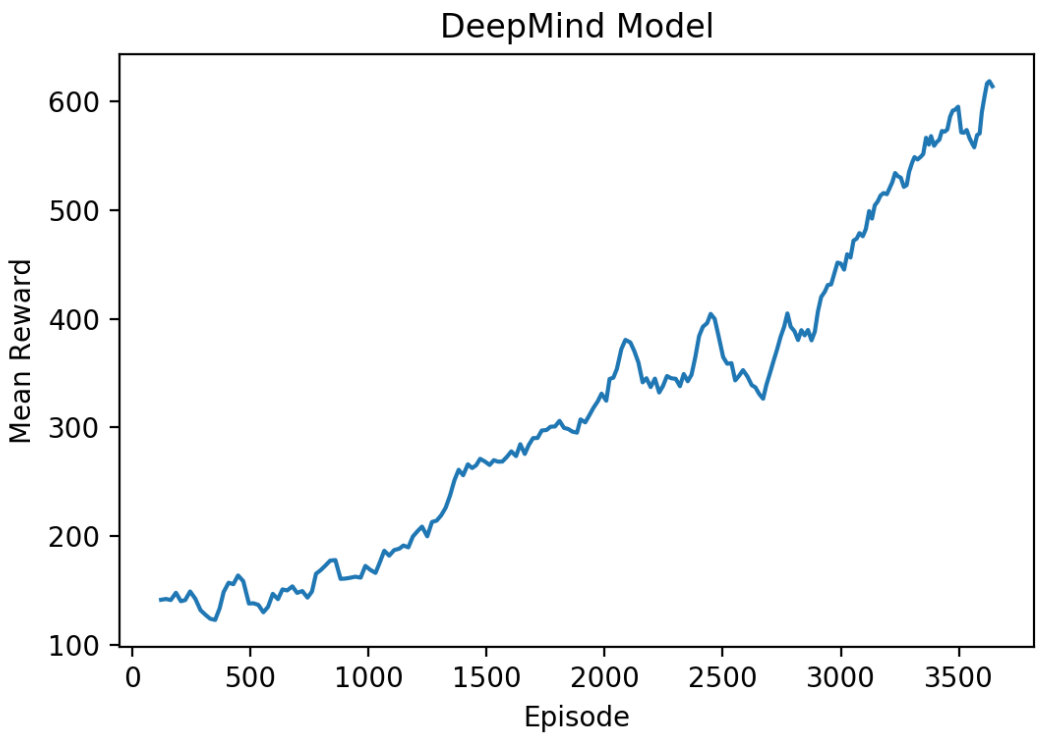
名称	描述
conv1	卷积层，深度为32，卷积核大小为8x8，步长为4，激活函数为ReLU
conv2	卷积层，深度为64，卷积核大小为4x4，步长为2，激活函数为ReLU
conv3	卷积层，深度为64，卷积核大小为3x3，步长为1，激活函数为ReLU
faltten	将卷积的结果展平为一维张量
V_fc1	Value函数的全连接层，以flatten为输入，节点数为512，激活函数为ReLU
Value	Value函数的结果，节点数为1
A_fc1	Advantage函数的全连接层，以flatten为输入，节点数为512，激活函数为ReLU
Advantage	Advantage函数的结果，节点数为4（合法动作的数量）
out	q_func的结果，综合考虑了Value与Advantage的结果

模型训练结果

以下选取两次结果（分别为DQN与Dueling DQN模型，其他结构与参数相同，参考了DeepMind的论文）进行说明

DeepMind 原始模型

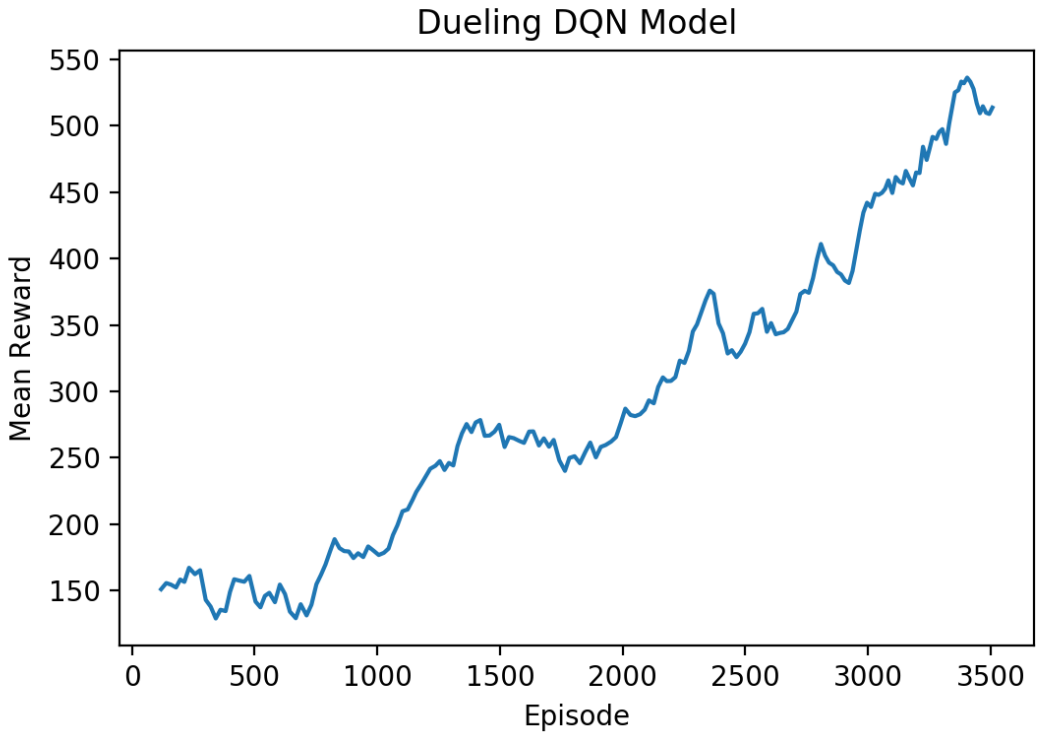
> Reward-episode图像



在3500个episode的训练后，模型的平均reward（游戏分数）由150提升到600+

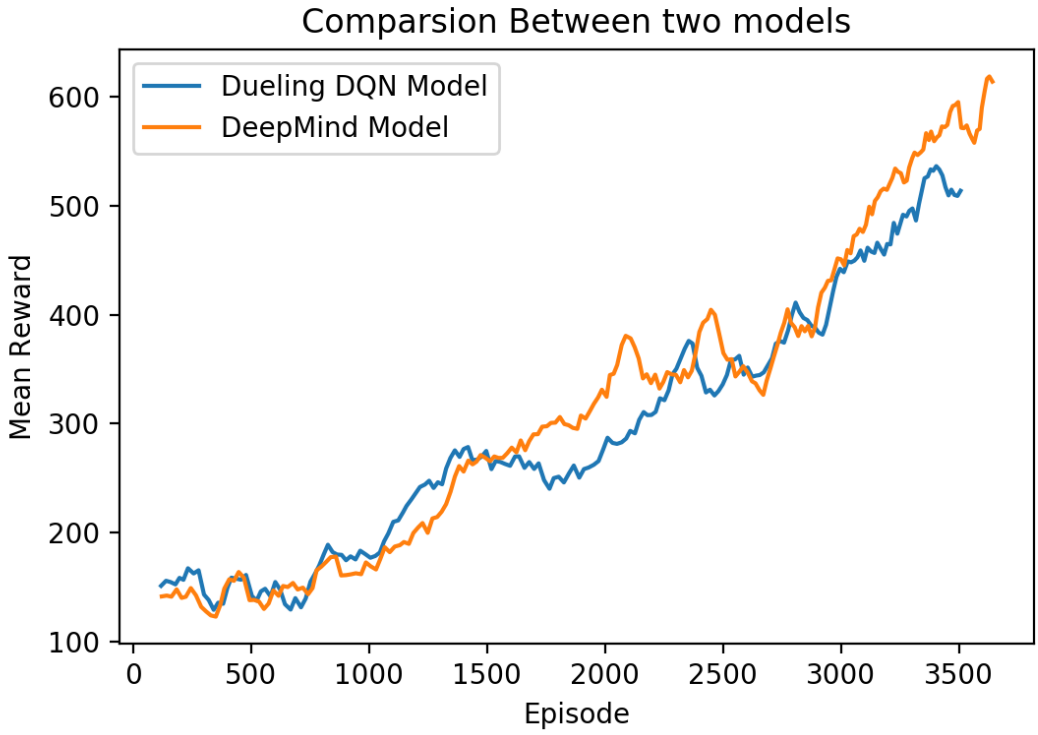
Dueling DQN 模型

> Reward-episode图像



在3500个episode的训练后，模型的平均reward由150提升到500+

模型对比



可以看到，两个模型的表现并没有显著的不同。一定程度上可能是由于Dueling DQN模型需要训练的参数个数多余DQN模型，导致训练的难度加大。

To be continued