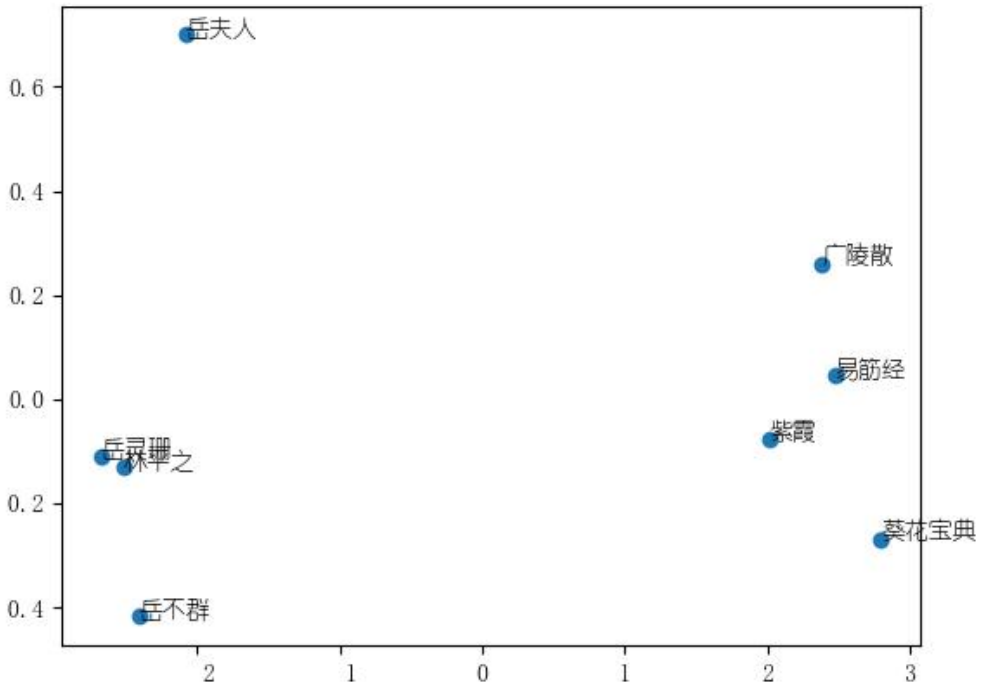
文本表示学习实验

一、 word2vec 词表示

（1） 截图与岳不群最相近的十个词、截图人物与功夫在空间中的相对位置

```
--与<岳不群>最相似的10个词--
林平之 0.9859698414802551
岳灵珊 0.9835317134857178
任我行 0.9797444939613342
向问天 0.9796097874641418
岳夫人 0.9702379703521729
盈盈 0.9702238440513611
林震南 0.9592429399490356
劳德诺 0.9573549032211304
刘正风 0.9571712613105774
方证 0.9569605588912964
```



（2） 给出重新训练后下面两个词的相似度 sim(岳不群, 林平之)和 sim(岳不群, 岳林珊)

```
sim(岳不群，林平之)：  0.96762204
sim(岳不群，岳灵珊)：  0.9556633
```

（3） 截图给出与岳不群 + 令狐冲 - 岳夫人最相似的 5 个人物

训练前：

```
--和<岳不群+令狐冲-岳夫人>最相似的词--
盈盈 0.9045735001564026
岳灵珊 0.9043087363243103
林平之 0.9003630876541138
余沧海 0.8730987310409546
向问天 0.8669897317886353
```

训练后：

```
--和<岳不群+令狐冲-岳夫人>最相似的词--
盈盈 0.9215347170829773
余沧海 0.9213935136795044
林平之 0.9179047346115112
东方不败 0.8995457887649536
任我行 0.8969937562942505
```

（4） 给两对相似词的相似度/两对不相似词的相似度

a） 如 sim(岳不群，令狐冲)=0.95 sim(岳不群，吸星大法)=0.5

这里使用人民日报语料进行训练，结果如下：

```
sim(实事求是，解放思想): 0.9711194038391113

sim(法国，新年): 0.7006419897079468
```

二、 LSTM 搭建与训练

（1） 提交 rnn.py，其中应该完整包括完整的数据处理与模型代码以及训练代码

（2） 截图训练的 loss

```
Epoch 10, Iteration 1100, Loss: 2.794570207595825
Epoch 10, Iteration 1200, Loss: 3.0667314529418945
Epoch 10, Iteration 1300, Loss: 2.885589838027954
Epoch 10, Iteration 1400, Loss: 2.621288299560547
Epoch 10, Iteration 1500, Loss: 2.7346935272216797
Epoch 10, Iteration 1600, Loss: 2.6765198707580566
Epoch 10, Iteration 1700, Loss: 2.948638677597046
Epoch 10, Iteration 1800, Loss: 2.8226561546325684
Epoch 10, Iteration 1900, Loss: 2.1852376461029053
Epoch 10, Iteration 2000, Loss: 2.5089776515960693
Epoch 10, Iteration 2100, Loss: 2.4727895259857178
Epoch 10, Iteration 2200, Loss: 3.022475242614746
Epoch 10, Iteration 2300, Loss: 2.4793009757995605
Epoch 10, Iteration 2400, Loss: 2.9342458248138428
Epoch 10, Iteration 2500, Loss: 2.306018590927124
Epoch 10, Iteration 2600, Loss: 2.4219300746917725
Epoch 10, Loss: 2.6251351666745286
```

三、 基于 BERT 的词表示与句子表示

（1） 给出 BERT 的参数计算过程，如

"hidden_size": 768,

"intermediate_size": 3072,

"max_position_embeddings": 512,

"num_attention_heads": 12,

"num_hidden_layers": 12,

"position_embedding_type": "absolute",

"vocab_size":

a) 词向量为 30522*768，总参数为 23,440,896

b) 位置编码为 512*768，总参数为 393,216

c) 一层 transformers 参数计算：

i. 多头自注意力：(768 * 768 * 12 / 12 * 3 + 768 * 768) = 2,359,296

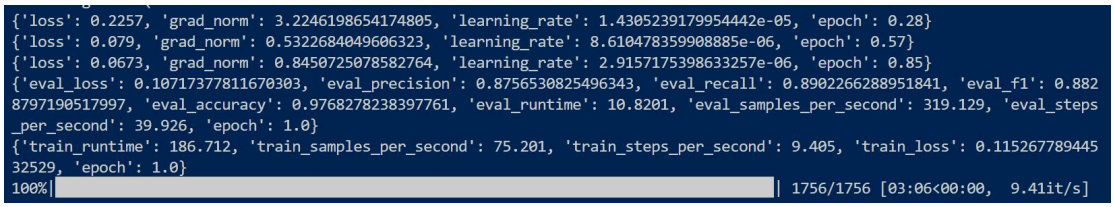
ii. 前馈神经网络：768 * 3072 * 2= 4,718,592

iii. 层归一化：hidden_size*2*2=768*2*2=3072；

d) 总参数为： 23,440,896+393,216+12*（2,359,296+4,718,592+3072）

=108,805,632

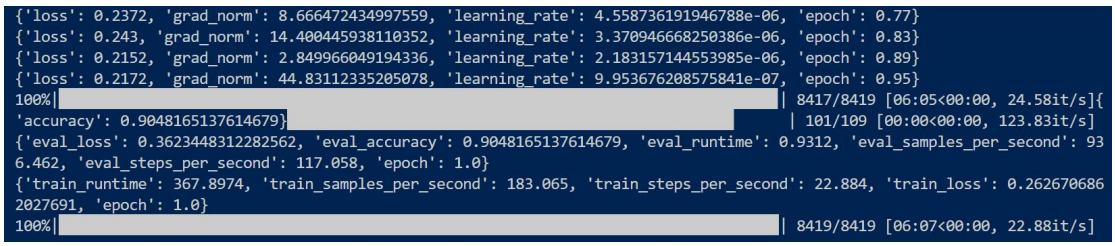
（2） 截图训练结束之后的评估结果



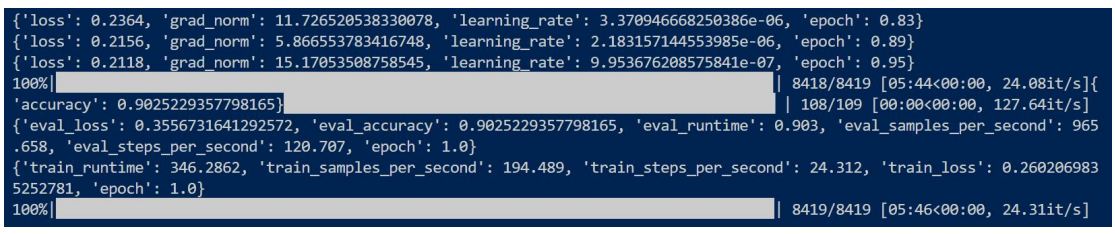
（3） 提交 train_cls.py，其中应该包括完整的训练代码。提交 BERT.py，其中包括完整的 mean pooling 代码与 max pooling 代码，仅提交一个 BERT.py 文件，其中不同的 pooling 代码注释掉。补充下面的不同方法的性能的表格，其中使用 CLS 的方法 accuracy 需要大于 0.9：

方法	Accuracy
CLS	0.9048165137614679
Mean pooling	0.9025229357798165
Max pooling	0.8956422018348624

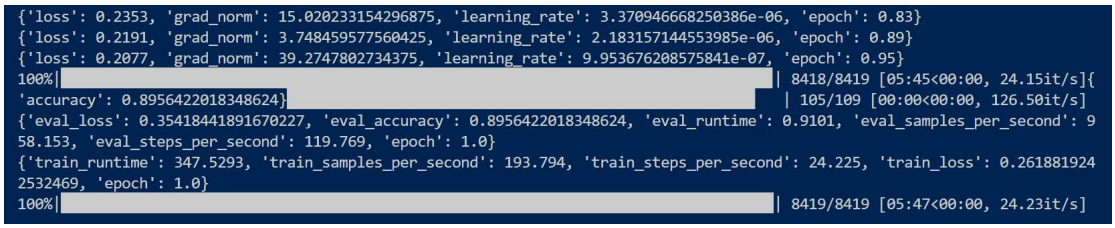
CLS：



Mean pooling：



Max pooling:


{'loss': 0.2353, 'grad_norm': 15.020233154296875, 'learning_rate': 3.370946668250386e-06, 'epoch': 0.83}
{'loss': 0.2191, 'grad_norm': 3.748459577560425, 'learning_rate': 2.183157144553985e-06, 'epoch': 0.89}
{'loss': 0.2077, 'grad_norm': 39.2747802734375, 'learning_rate': 9.953676208575841e-07, 'epoch': 0.95}
100%| 8418/8419 [05:45<00:00, 24.15it/s]{
'accuracy': 0.8956422018348624} | 105/109 [00:00<00:00, 126.50it/s]
{'eval_loss': 0.35418441891670227, 'eval_accuracy': 0.8956422018348624, 'eval_runtime': 0.9101, 'eval_samples_per_second': 9
58.153, 'eval_steps_per_second': 119.769, 'epoch': 1.0}
{'train_runtime': 347.5293, 'train_samples_per_second': 193.794, 'train_steps_per_second': 24.225, 'train_loss': 0.261881924
2532469, 'epoch': 1.0}
100%| 8419/8419 [05:47<00:00, 24.23it/s]