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

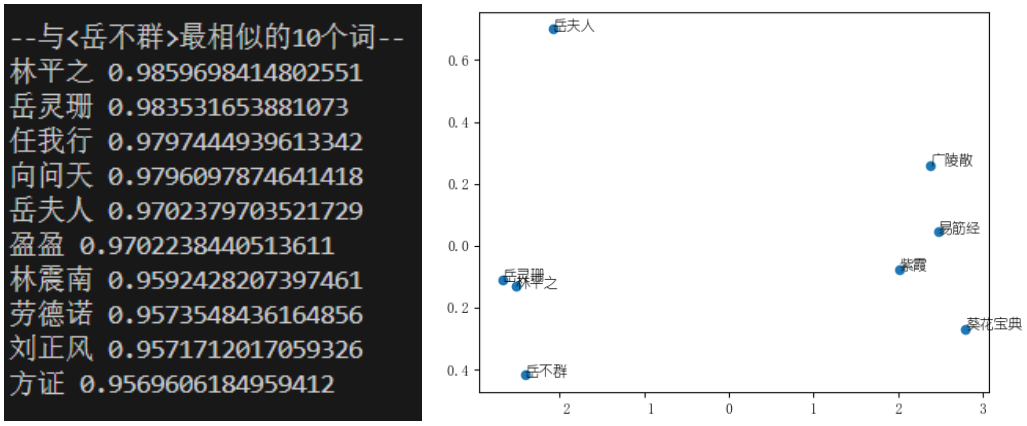

图 1 结果截图（左：与岳不群最相近的十个词；右：人物与功夫在空间中的相对位置）

（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)

此步骤主要尝试修改窗口大小↓、嵌入向量维度↑、训练轮次参数↓来提高语义相似度，减少训练轮次目的是让模型训练不充分欠拟合，提高嵌入向量维度的目的是提高模型的复杂度，降低窗口大小则是为了限制模型对上下文复杂语义特征的学习。

以上方法均使得模型不能正确的学到词之间的联系和语义，仅从最简单的词性等角度来评估单元之间的相似性，从而提高词之间的相似度，具体结果如下所示：

1. **设置窗口大小为 3，嵌入向量维度为 64，训练轮次为 1:**
   - sim(岳不群, 岳灵珊) = 0.9745843410491943
   - sim(岳不群, 林平之) = 0.9592603445053101
2. **设置窗口大小为 3，嵌入向量维度为 512，训练轮次为 1:**
   - sim(岳不群, 岳灵珊) = 0.9827371835708618
   - sim(岳不群, 林平之) = 0.9782577157020569
3. **设置窗口大小为 1，嵌入向量维度为 64，训练轮次为 3:**
   - sim(岳不群, 岳灵珊) = 0.9548320174217224
   - sim(岳不群, 林平之) = 0.9543564915657043
4. **设置窗口大小为 1，嵌入向量维度为 512，训练轮次为 3:**
   - sim(岳不群, 岳灵珊) = 0.9633985757827759
   - sim(岳不群, 林平之) = 0.9624267220497131

（3） 截图给出与岳不群 + 令狐冲 - 岳夫人最相似的 5 个人物

以仅修改训练轮次为 1 的情况为例（第 1 种），和基于 CBOW 的方法对比如下：

图 2 结果截图（左：基于 CBOW 训练的方法；右：修改后基于 skip-gram 方法模型）

（4） 给两对相似词的相似度/两对不相似词的相似度

使用小说《斗破苍穹》作为新语料，在窗口大小为 3，嵌入向量维度为 64，训练轮次为 3 的训练设置下训练结果如下：

a) sim(萧炎，古元)= 0.8483；sim(萧炎，融血丹)= 0.2954
b) sim(焚决，弄焰决)= 0.8037；sim(焚决，云韵)= 0.2801
c) sim(斗尊，斗宗)= 0.9594；sim(斗尊，萧炎)= 0.4170

注：萧炎、云韵、古元为人物名，焚决、弄焰决为功法，斗尊、斗宗为等级，融血丹为丹药

二、 LSTM 搭建与训练

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

1. 数据处理部分

读取文本数据并使用 jieba.cut() 函数对文本进行分词处理，而后使用 Counter 对分词后的词语列表进行计数，得到各个词语的出现次数，选取出现次数最多的 vocab_size - 1 个词语作为词表，并将余下的词(out-of-vocabulary，OOV)设定为 <unk>。

将分词后的词语列表 words 中的每个词语根据词汇表转换为对应的索引。之后在构建数据集的过程中，每个样本是一个输入序列和对应的输出序列，其中输出序列是输入序列向右移动一个位置，用于语言建模任务，根据前面的词语预测下一个词语。

2. 模型部分

模型部分包括一个嵌入层、一个 LSTM 层，以及一个线性层(语言建模头)。

a) 嵌入层的作用是将稀疏的词语的索引转换为密集的嵌入向量。
b) LSTM 层处理输入序列的长期依赖关系，捕捉文本数据中的语义信息。
c) 线性层是用于将 LSTM 层的输出映射回词汇表大小的向量，预测下一个词语的概率分布。

在初始化函数中定义了一个用于初始化隐藏状态的方法 init_hidden。隐藏状态在 LSTM 中起到了记忆过去信息的作用，而初始化隐藏状态是为了在模型开始预测新的序列时，将其重置为初始状态。

3. 训练部分

选用交叉熵损失函数，用于优化生成的语言分布。采用 Adam 优化器进行优化，当模型 Loss < 3 时停止训练。

（2） 截图训练的 loss

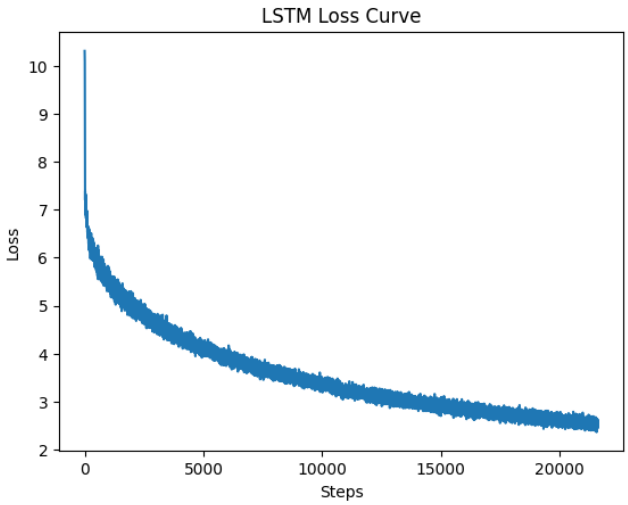

图 3 训练过程 loss 变化

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\mr\AppData\Local\Temp\jieba.cache
Loading model cost 0.520 seconds.
Prefix dict has been built successfully.
Epoch 1: 100%| | 10796/10796 [14:06<00:00, 12.76it/s, loss=3.29]
Epoch 1, Loss: 3.2853879928588867
Epoch 2: 100%| | 10796/10796 [13:58<00:00, 12.87it/s, loss=2.47]
Epoch 2, Loss: 2.468625783920288
```

图 4 LSTM 训练结果

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

1. Embedding 部分

a) word_embeddings 的形状为[30522, 768]，参数量为$30522 \times 768 = 23440896$
b) position_embeddings 的形状为[512, 768]，参数量为$512 \times 768 = 393216$
c) LayerNorm 的形状为[768]，包含 Weight 和 Bias，参数量为$768 + 768 = 1536$
d) token_type_embeddings 的形状为[2, 768]，参数量为$2 \times 768 = 1536$

2. Transformer 部分

bert-base-uncased 模型总共包含 12 个 Transformer Block，每层 Transformer Block 包含注意力矩阵 query、key、value 和 output，中间层 intermediate，输出层 output，并应用层归一化，具体参数如下：

a) Attention 部分包含 query、key、value 和 output 四个部分，形状均为 Weight [768, 768]和 Bias[768]，参数量为$768 \times 768 + 768 = 590592$；同时 output 还包含一个 LayerNorm 层，形状为 Weight [768]和 Bias[768]，参数量为$768 + 768 = 1536$

b) intermediate 部分形状为 Weight [3072, 768]和 Bias[3072]，参数量为 $3072 \times 768 + 3072 = 2362368$

c) output 部分（和 attention 的 output 不同）形状为 Weight [768, 3072]和 Bias[768]，参数量为 $768 \times 3072 + 768 = 2360064$；同时还包含一个 LayerNorm 层，形状为 Weight [768]和 Bias[768]，参数量为 $768 + 768 = 1536$

因此，Transformer 部分参数量总共为：

$$(590592 \times 4 + 1536 + 2362368 + 2360064 + 1536) \times 12 = 85054464$$

3. Classifier 部分

仅包含一个线性层 Weight [2, 768]和 Bias[2]，总参数为 $2 \times 768 + 2 = 1538$

因此总的参数量为上述之和，为 $108893186(108M)$

（2） 截图训练结束之后的评估结果

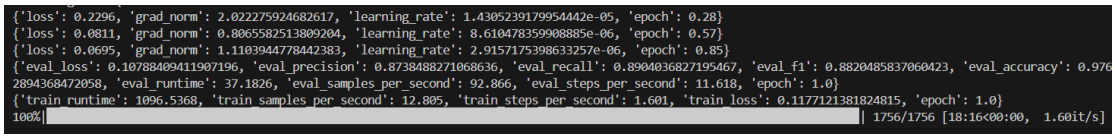

图 5 命名实体识别训练截图（词分类）

（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:

对于表示向量选择，可以选择[CLS] token、Mean pooling、Max pooling 方式，具体来说就是对 hidden_state 的第二个维度进行相关的操作(BERT.py 1008~1011 行)：

a) pooled_output = hidden_state[:, 0] # [CLS] token only

b) pooled_output = hidden_state.mean(dim=1) # Mean pooling

c) pooled_output, _ = hidden_state.max(dim=1) # Max pooling

在训练超参数设置为 batch_size=16，learning_rate=2e-5，epochs=1 下，最后训练得到的模型在 validation 验证集上的 Accuracy 如下：

方法	Accuracy
CLS	0.9071100917431193
Mean pooling	0.9013761467889908
Max pooling	0.8990825688073395

实验结果可以发现，三种表示方式并无明显性能差异，可能是数据集较为简单，对模型特征提取的能力要求不高，详细截图如下：

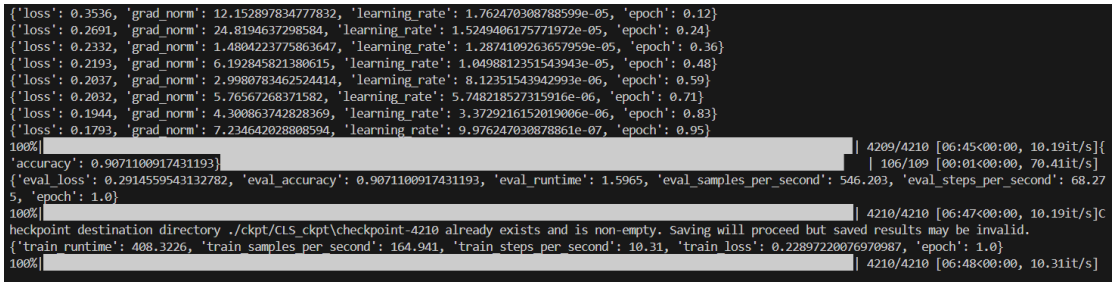

图 6 情感分类识别训练截图（使用 CLS Token）

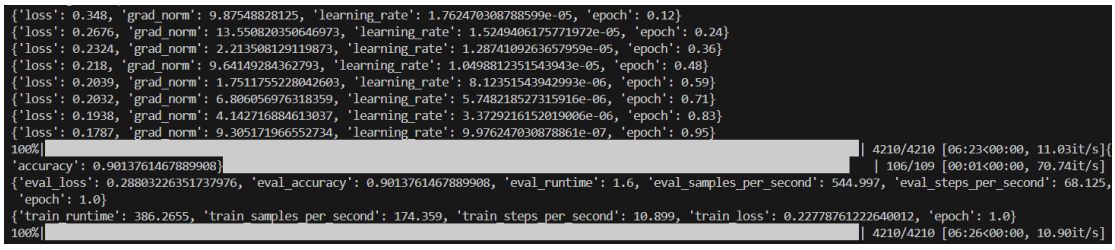

图 7 情感分类识别训练截图（使用 Mean pooling）

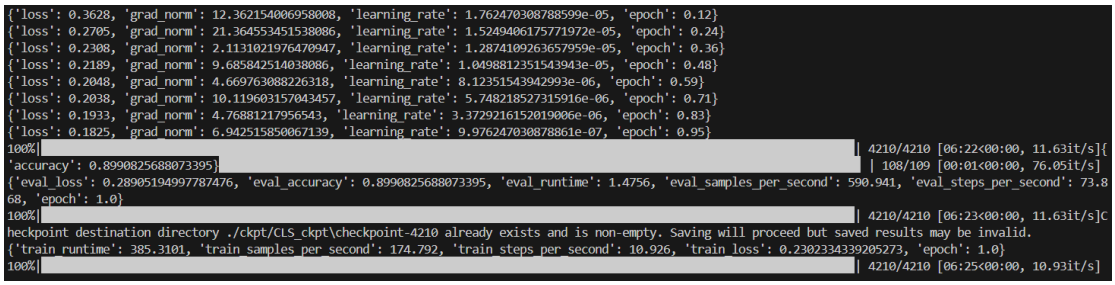

图 8 情感分类识别训练截图（使用 Max pooling）