文本表示学习实验

Word2Vec词表示

本次实验基于gensim库来探索表示学习方法，表示学习方法主要分为Skip-Gram和CBOW，其中基于CBOW算法基于窗口内的词来预测中间词，而Skip-Gram基于中间词预测周围词，基于优化算 法训练之后，中间的映射矩阵就是我们所需要的词向量矩阵，可以用于表示每一个词，实验证明词向量 中包含着非常丰富的信息。

实验环境

本次实验基于python，首先需要自行安装anaconda库，然后安装以下库：

anaconda安装可以参考https://zhuanlan.zhihu.com/p/459601766

```
pip install gensim
pip install scikit-learn
pip install matplotlib
pip intall jieba
```

实验要求

- 1.运行`play.py`基于CBOW训练词向量，并分别得到相似词列表、人物与功夫在二维空间中的相对位置。查看与**岳不群**最相近的五个词，查看人物与功夫在二维空间中的相对位置。
- 2.修改`word2vec.py sg=1`，使用`skip-gram`方法，并调整参数，重新运行`word2vec.py`训练模型，使得相似度：
  - `sim(岳不群，林平之)>0.95`
  - `sim(岳不群，岳林珊)>0.95`
- 3.查看人物在空间中的相对位置关系，给出与**岳不群 + 令狐冲 - 岳夫人** 最相似的人物
- 4.至少选择一个新的语料，可以是小说、百科等，使用`process.py`处理文件，然后在该语料进行训练，测试并给出2对相似词的相似度和2对不相似词的相似度（比如人物和人物相似而人物与功夫不相似）

LSTM搭建与训练

实验环境

```
pip install torch
```

实验要求

- 1.基于`pytorch`搭建`LSTM`

- 2.使用`./data/text.txt`中的语料，搭建模型训练框架，对模型训练模型并保存，模型训练目标为语言建模任务(Language Modeling)
- 3.使用交叉熵(CrossEntropyLoss)的`loss < 3`

  单层为词表大小为30000，嵌入层为64维的LSTM，训练最大长度为31，在该语料上大约需要10分钟训练一轮，可以视环境进行模型参数量配置

基于BERT的词表示与句子表示

实验环境

```
set TRANSFORMERS_OFFLINE=1 //linux为export TRANSFORMERS_OFFLINE=1
pip install transformers
pip install safetensors
pip install datasets
pip install accelerate -U
pip install evaluate
pip install seqeval
```

实验要求

- 1.已知 bert-base-uncased的参数量大小约为110M，并且其的设置如下：

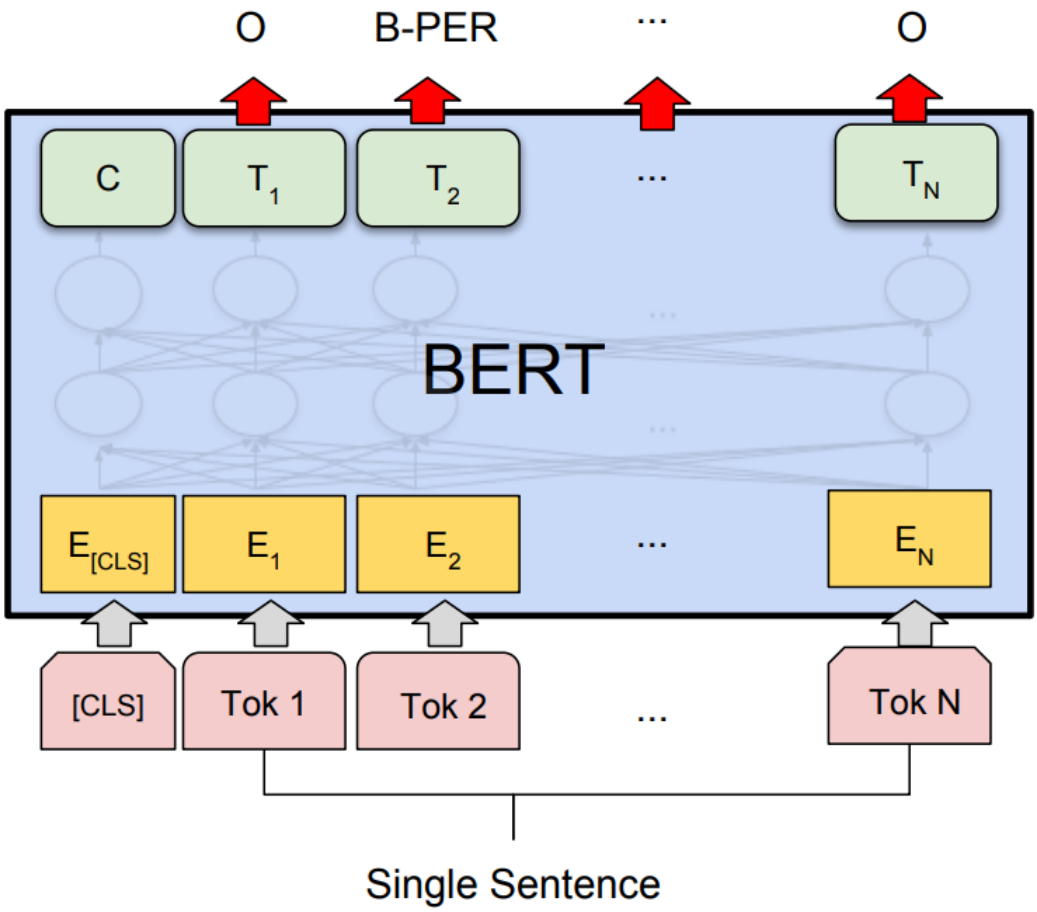
```
  "hidden_size": 768,
  "intermediate_size": 3072,
  "max_position_embeddings": 512,
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "position_embedding_type": "absolute",
  "vocab_size": 30522
```

详细给出该模型的参数计算过程。

- 2.基于BERT进行词分类，完成NER任务NER（命名实体识别）是经典的词分类任务，旨在为一句话中的每一个词打标签，常用的标签有

```
O   Outside of a named entity
B-MISC  Beginning of a miscellaneous entity right after another miscellaneous
entity
I-MISC  Miscellaneous entity
B-PER   Beginning of a person’s name right after another person’s name
I-PER   Person’s name
B-ORG   Beginning of an organization right after another organization
I-ORG   organization
B-LOC   Beginning of a location right after another location
I-LOC   Location
```

基于BERT的词分类结构如下：



对于输入句子的中的每一个词，其对应的输出向量可以作为该词的**表示**，该表示经过线性层映射到对应的分类类别，从而完成分类过程。

例如对于句子**I live in China**

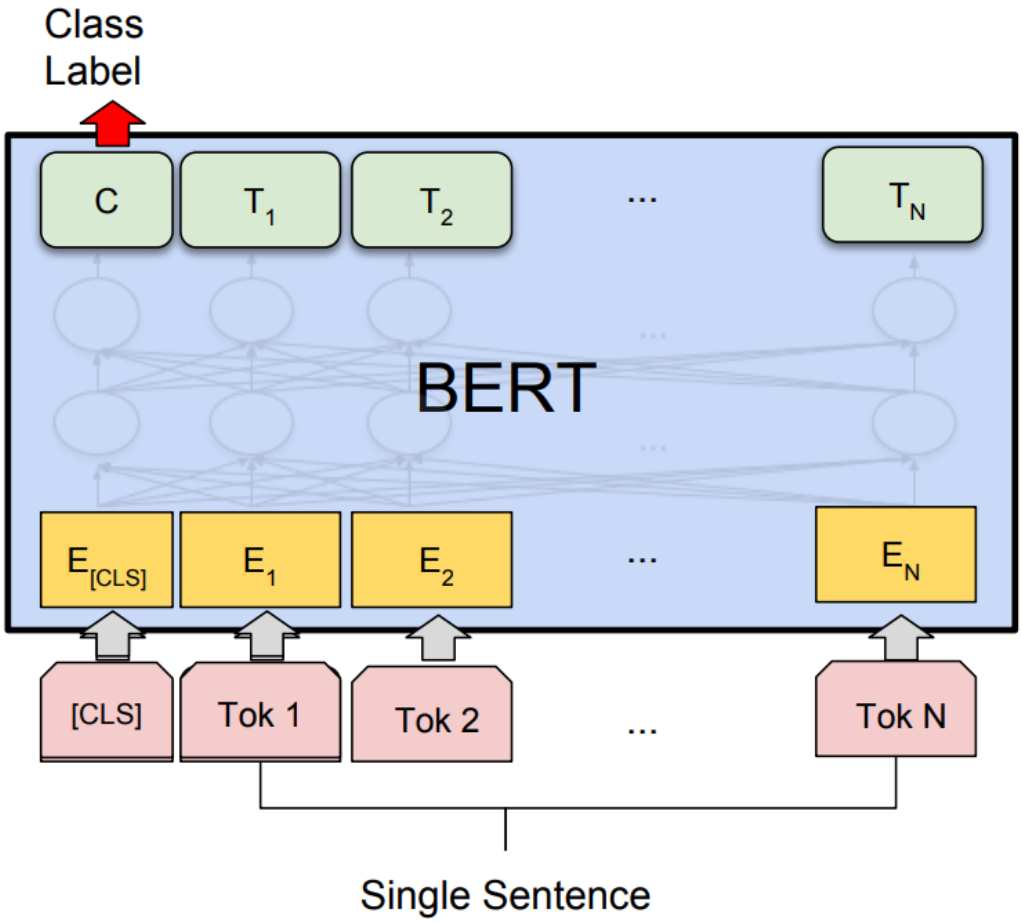
其标签如下为**O O O B-LOC**

阅读`train_ner.py`中的代码，运行该代码完成模型的训练与评估，并达到指标：

```
Precision > 0.925
Recall > 0.93
F1 > 0.93
Accuracy > 0.983
```

在10代5上，完成一轮训练大约需要1小时

- 3.基于BERT完成句子分类，情感分类是常见的句子分类任务，旨在给一句话打标签，可以评估模型的句子表示能力。基于BERT的情感分类结构如下：



对于BERT的输入，如`I live in China`在经过`tokenizer`之后，会变成`["CLS", "live", "in", "China", "[SEP]"]`的形式，其中头尾的`token`是BERT在预训练中使用的特殊字符，可以用于标注句子的开始结束/或者作为相邻句子的分隔符，我们往往使用`[CLS]`的输出向量作为模型输出的代表向量，并经过线性层，将输出结果作为分类结果。

对于表示向量选择，代表性的有三种：

- **CLS**：选择[CLS]标签的输出向量作为一句话的整体向量表示
- **Mean pooling**：将一句话的每个词的输出向量进行平均作为该句子的向量表示
- **Max pooling**：将一句话的输出向量中，每个维度最大的值拼成一个向量，作为该句子的向量表示

（1）.补充`train_cls.py`，完成模型的训练与评估流程，在`validation`验证集上达到`accuracy > 0.9`

（2）.修改`BERT.py`(1007行使用了`CLS`作为表示向量)，使用`mean pooling`，完成模型的训练与评估流程，测试在验证集上的性能并于对比其他向量选择方法

（3）.修改`BERT.py`，使用`max pooling`，完成模型的训练与评估流程，测试在验证集上的性能，并对比其他向量选择方法

在10代5上，完成一轮训练大约需要1.5小时