参考文献：

<https://kexue.fm/archives/8069>

<https://kexue.fm/archives/8321>

https://arxiv.org/abs/2103.15316

tensorflow版

<https://github.com/bojone/BERT-whitening>

Pytorch版

<https://github.com/autoliuweijie/BERT-whitening-pytorch>

这个是关于tokenizer的解释

<https://blog.csdn.net/pearl8899/article/details/119328276>

关于池化的解释

<https://blog.csdn.net/choushi5845/article/details/100747117>

关于bert输出的解释

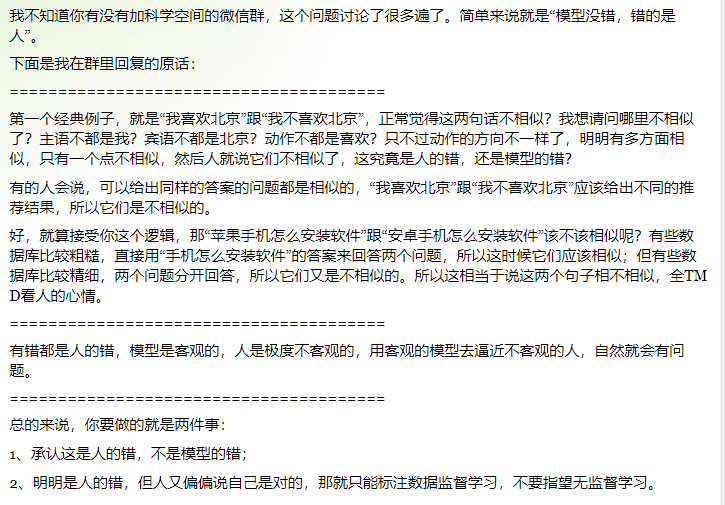
<https://www.cnblogs.com/xiximayou/p/15016604.html>

关于bert输入的解释

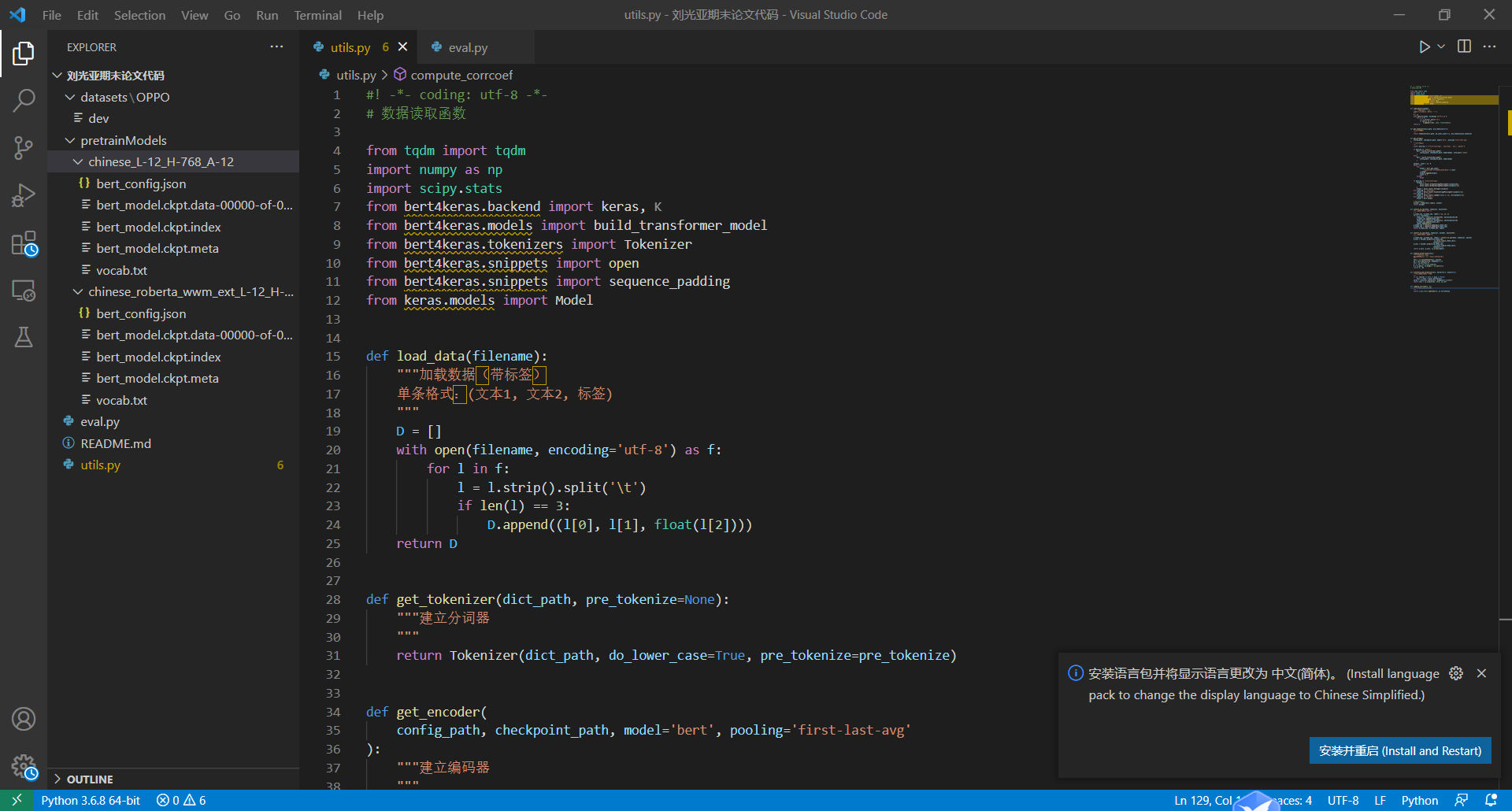
<https://www.cnblogs.com/gczr/p/11785930.html>

内含seq2seq的详解

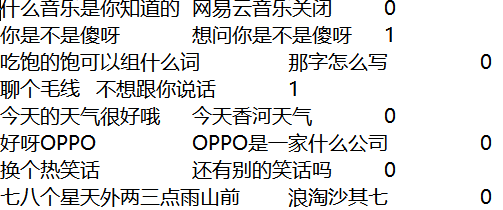
https://www.cnblogs.com/gczr/p/11785930.html



目录结构



数据集样例



做的方法有两种：

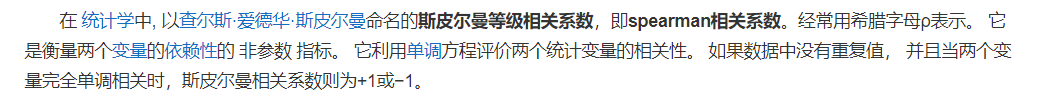
1第一种，也就是有监督，那么这个问题就会变成输入两个句子（拼接输入到bert）里，然后分类（0或1）。

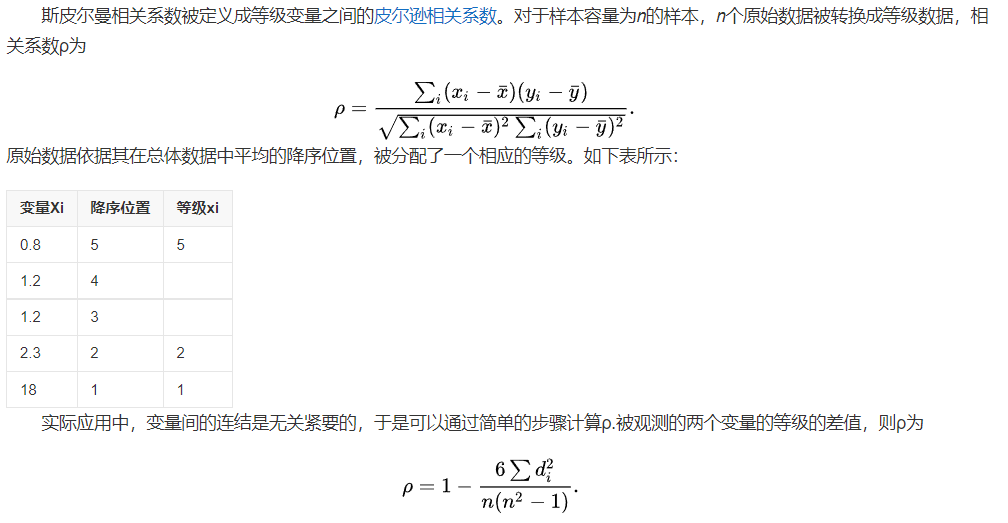
2第二种，无监督，.也就是说直接将句子a和句子b分别用向量表达，计算他们的相似度，得到在0和1之间的的某个相似度。这个代码用的是第二种，也就是无监督。

这种任务有什么用呢？在搜索引擎里很有用。节约空间（不需要存储那么多的qa对），提高速度（直接利用已有的qa的答案）。

评测标准

都是用spearman相关系数，这是一个类似AUC的排序指标，只依赖于预测分数的顺序，并且不依赖于阈值，所以用来评测效果是比较适合的。之所以没有用大家更熟悉的准确率（accuracy）指标，一是因为准确率依赖于具体的阈值，二是因为STS-B数据集的标签是1～5的数字，并不是0/1标签，要算准确率也无从算起，因此统一用spearman相关系数了。如果读者非要从准确率角度理解，那么大概可以认为“accuracy ≈ 0.5 + spearman / 2”吧。





环境：

Google colab gpu

tensorflow 1.14 + keras 2.3.1 + bert4keras 0.10.5，时间应该在1分钟左右

X 0.1 0.2 0.3

Y 0 1 0

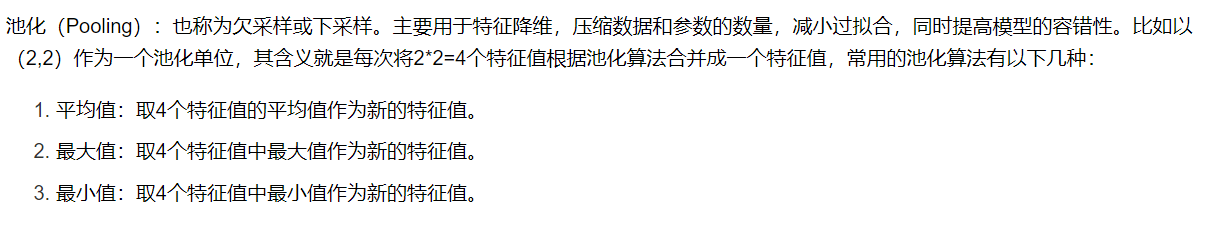
整体逻辑是

Baseline利用bert直接得出向量，然后计算相似度。

改进就是计算出向量后，利用他说的白化的方法对向量进行一些变换。

1. 先读数据 数据格式是 [[句子a, b,标签]]
2. 然后tokenizer把他encode，这个时候是 word 2 ids 也就是 “我”这个字在任何句子里面都是同一个id，本质上就是一个查字典 即tokenizer.encode()
3. 然后encoder把id做embedding，即encoder.predict()例如

之后选择池化方式输出向量即可，这里采用了不同的pooling方式，例如直接用cls代表句子或者最后一层每一个token的向量的平均代表句子



论方法 对方法进行反思和讨论 方法论

bert的embedding可不可以还原