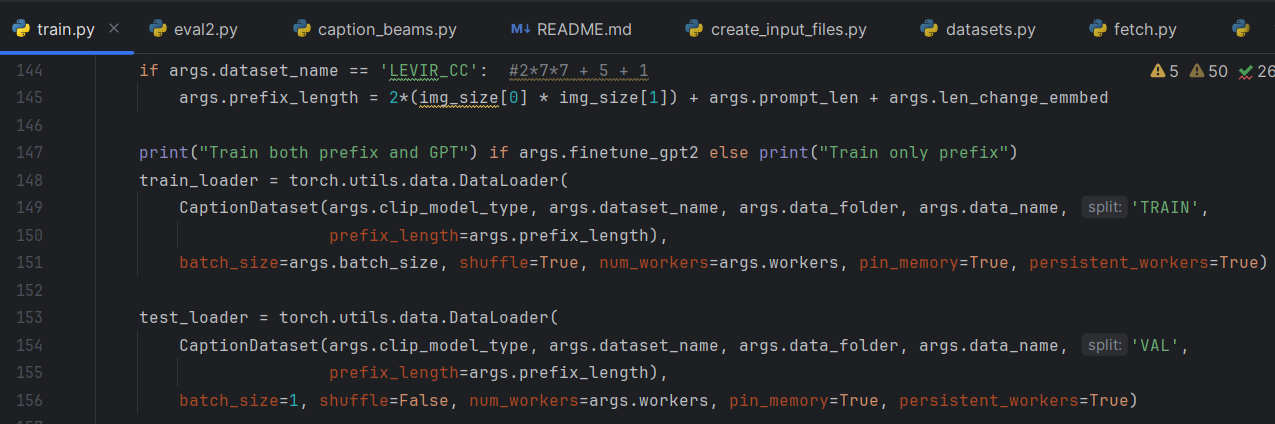
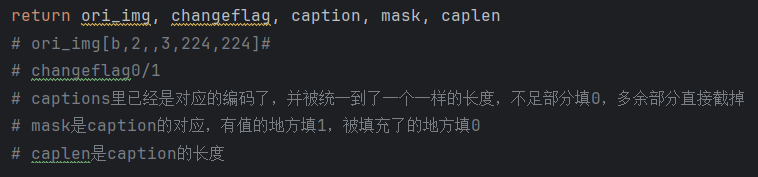
原文思路：





考虑batchsize=1，Dataloader返回：

一对图片[1,2,3,224,224]

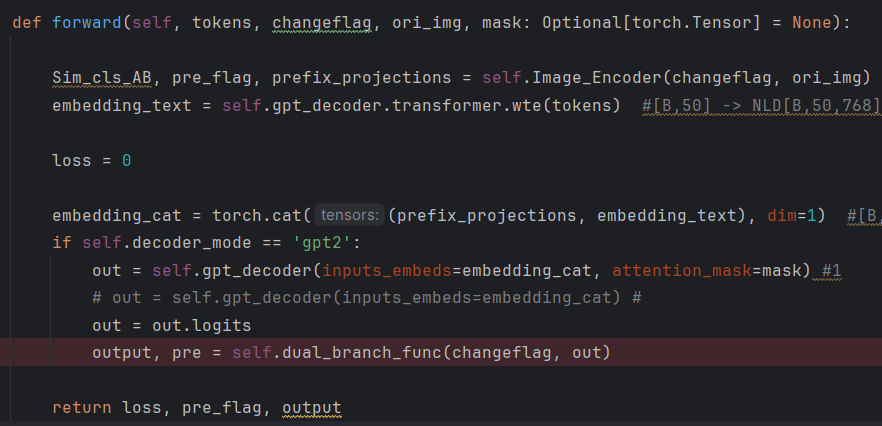
是否改变的changeflag

一个经过gpt后的语句（一对图片对应的是5个语句，为什么只返回一个语句，是因为这个dataset是根据语句找到对应图片的）

语句的mask

语句的长度

Dataloader的返回值进入模型的主体部分



Image\_Encoder返回：Sim\_cls\_AB为定值0，后面页没用到，不评价

Pre\_flag=0或1，是模型预测的是否有改变

Prefix\_projections是 图片经过处理之后形成的[b,98,768]的tensor和prompt合并之后的tensor，形状为[b,104,768]。这个prompt就是论文中提到的Pu和Pci，形状分别为[b,1,768]和[b,5,768].

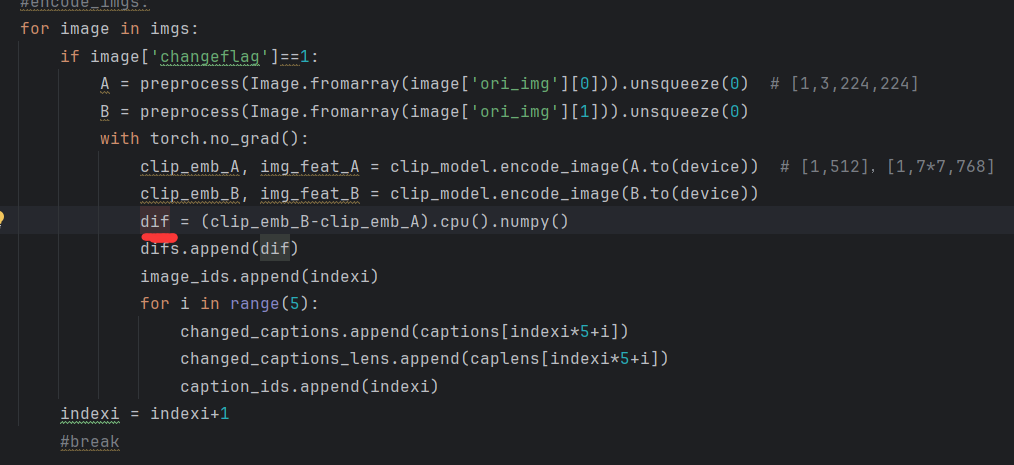
Embedding\_text对一个经过gpt后的语句进行encode，encode之后的形状为[b,50,768]

Prefix\_projections和Embedding\_text合并，形成[b,154,768]。这个综合了图片和语句的张量被丢进后续的模块，得到输出。

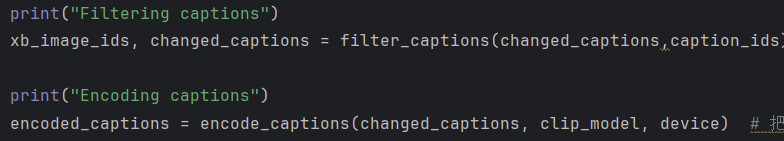
修改思路

先运行retrive\_caps.py

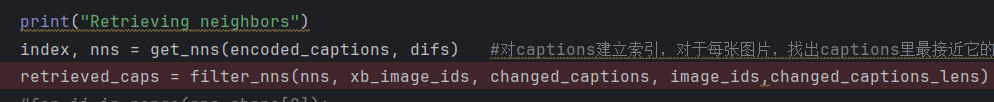
retrive\_caps.py



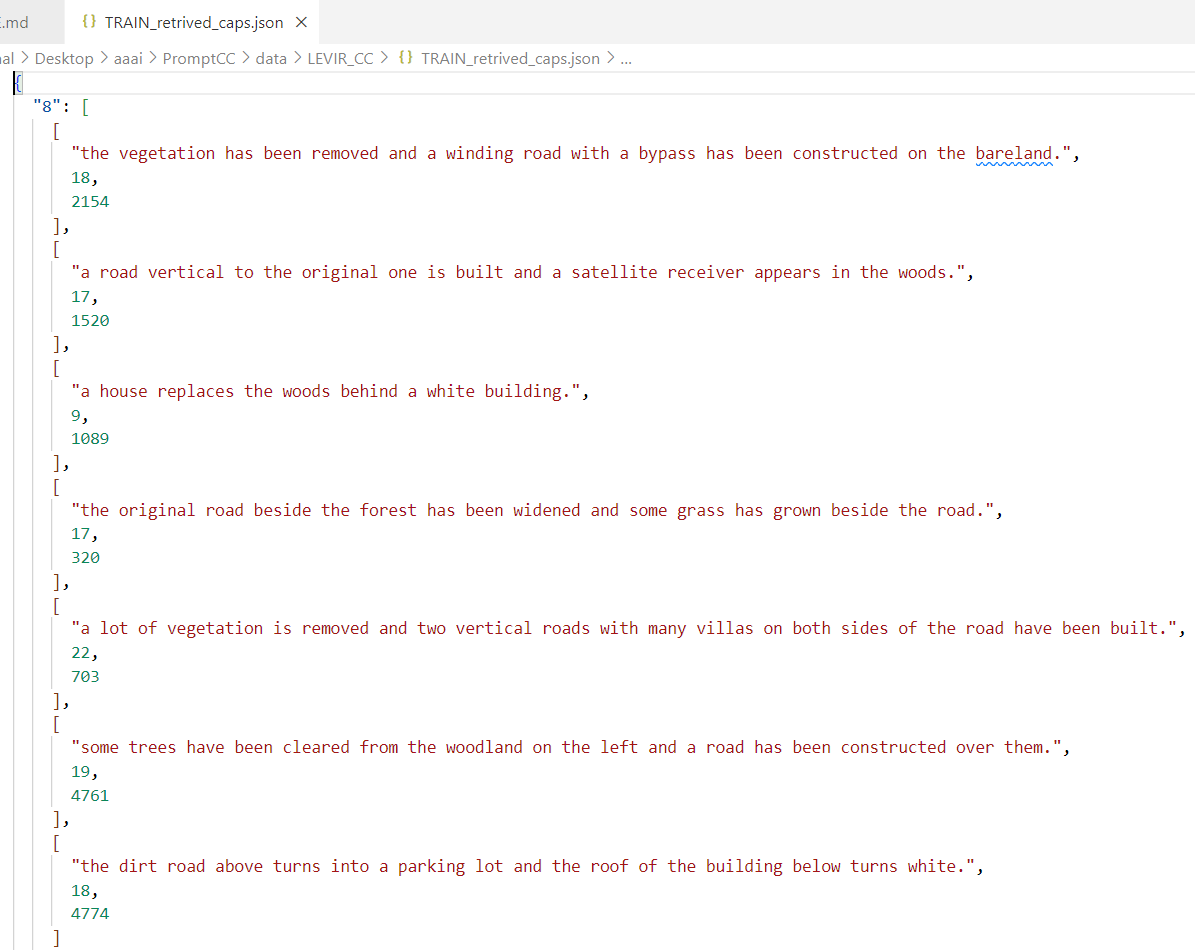
retrive\_caps先将changeflag=1的图片编码之后相减，得到difs



把changeflag=1的那些图片对应的语句筛选出来，并进行编码，得到encode\_captions。（即encode\_caotions里不包括那些changeflag=0的图片对应的语句）



对于每个dif，找与它最相近的k=7个encode\_caotions（k=7与smallcap保持一致）



最终写入的json文件中，’8’代表图片的编号，在’8’里面有k=7个list，每个list里面包含一个语句，语句的长度，语句对应的图片。

原先的数据集是 一个图片对对应五个语句。现在是一个图片对对应5+7=12个语句。

但在取数据的时候，由于dataloader是先取一个语句，再通过语句找对应的图片。

因此，一次性输入到模型的东西相当于只有一张图和一个语句。我们这么做只是把一张图对应的五个语句变成了12个语句，但是每次输入的东西还是只有一张图和一个语句。

所以现在是，我们输入的图片所对应的语句可能不是这张图的了（因为加了7个不是这张图的语句）