**IWSLT14 En-Zh机器翻译作业**

1. 实验目的
   1. 进一步加深对机器翻译任务基本目标和流程的理解。
   2. 掌握卷积神经网络、循环神经网络、Transformer等处理文本的各项技术。
   3. 加强对pytorch、tensorflow等深度学习框架的使用能力。
2. 实验要求
   1. 任选一个深度学习框架建立一个机器翻译模型，实现在IWSLT14 En-Zh数据集上进行的机器翻译。
   2. 按规定时间在课程网站提交实验报告、代码。
3. 任务介绍

标准的机器翻译流程包括：

* 1. 数据预处理及分词（BPE，JIEBA等）
  2. 生成模型的构建及训练（RNN, CNN, Transformer等）
  3. 生成模型的解码（Greedy, Beam Search等）

本作业主要针对生成模型的构建及训练部分进行考核，对分词方式和解码方式不做过多的要求。**序列生成任务需要大量的计算资源，公平起见，本作业会按照提交的代码判定成绩而不是模型的训练效果。**

1. 数据集介绍

本次实验旨在实现一个简单的中英翻译系统。所用的数据集是从小规模的机器翻译数据集IWSLT14 En-Zh中抽取的数据，其全部数据均在附件的en-zh.rar压缩包中。这个数据集包含了143920个训练样本，19989个验证样例和15992个测试样例。其中，训练集数据在train.zh/train.en中，验证数据在valid.zh/valid.en中，测试集数据在test.zh/test.en中。X.zh与X.en中的数据每一行是对齐的。

**如果计算资源有限，可以减小使用的数据比例。**

1. 参考模型

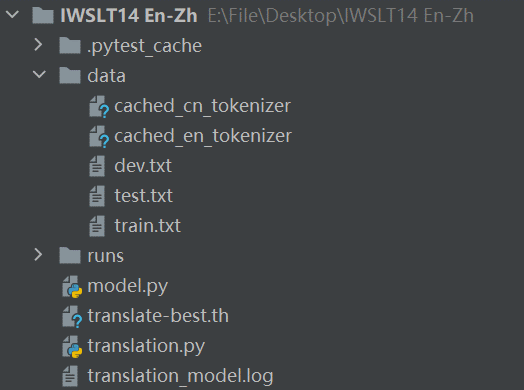
本次实验要求实现一种机器翻译模型即可，可采用**下面三篇参考论文中的任意一种或自己设计的结构**均可。

* 1. 使用CNN进行机器翻译，可参考2017年ICML的ConvS2S进行机器翻译的论文[[[1]](#footnote-1)]。
  2. 使用RNN/LSTM进行机器翻译，可参考2015年ICLR的基于LSTM+Attention的机器翻译论文[[[2]](#footnote-2)]。
  3. 计算资源比较充足的同学，也可采用Transformer进行机器翻译，可参考2017年NIPS的基于Transformer的机器翻译论文[[[3]](#footnote-3)]。

1. 作业提交说明
   1. 要求使用上述参考论文中的结构或自己实现的结构来实现机器翻译模型。
   2. 提交的作业除代码外，需包含一份简要报告来说明自己使用的分词策略、模型结构以及解码策略。如果有自己的一些独特的思考和实现可以详细写出。
2. 参考文献

**IWSLT14 En-Zh机器翻译报告**

1. 报告说明



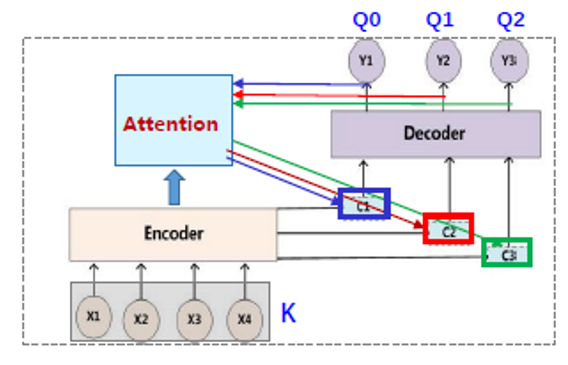
* Data目录为本次训练所使用的数据，为经过预处理后的数据，并且由于计算资源有限，保留了30000条训练样本，所以最后的模型效果有点不佳。如果使用原始的训练样本，一运行IDE和电脑直接就卡死，然后报爆内存的错误，然后连截图都截不了，只能重启。
* Runs目录用于保存每次训练记录
* translation\_model.log 用于记录训练过程。

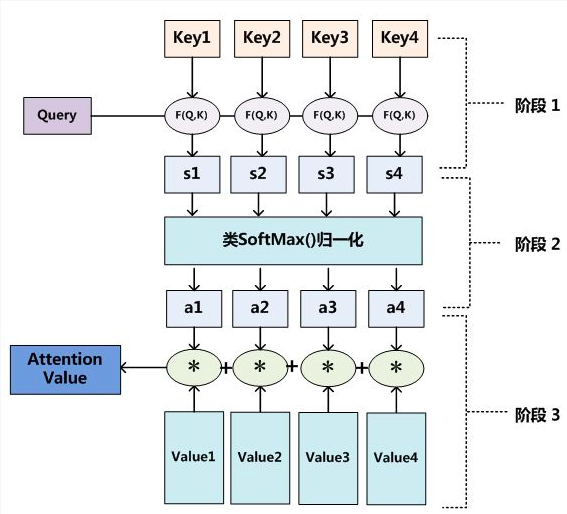


1. 模型基础

机器翻译，又称为自动翻译，是利用计算机将一种自然语言(源语言)转换为另一种自然语言(目标语言)的过程。翻译的三大核心要素：信、达、雅，不过不在我们考虑的范围之内。我们采用的是Seq2Seq模型。

Seq2Seq将一个序列映射成另一个序列，一般基于Encoder-Decoderm框架，Seq2Seq 属于 Encoder-Decoder 的大范畴但是Seq2Seq 更强调目的，Encoder-Decoder 更强调方法。并且加入Attention机制，如下图所示。



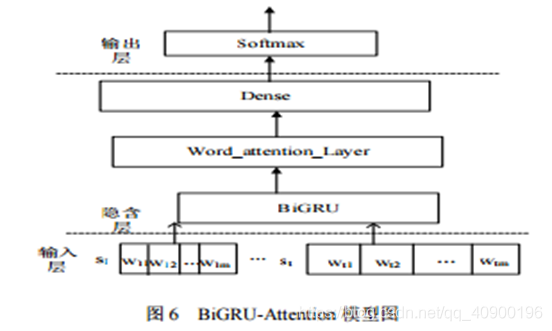


每输入一个词，encoder就更新一个状态，把输入的信息记录在encoder的状态h里，最后一个状态就是从英文中提取出的特征，encoder只输出最后一个状态，丢掉之前所有的状态。将最后一个状态传给decoder网络，将encoder的最后输出状态作为decoder的输入状态，这样decoder网络就知道输入的英文句子了。decoder网络相当于一个文本生成器，首先将起始符作为decoderRNN的输入，decoder RNN会更新状态为s1，全连接层输入预测的概率p1，根据概率分布p1抽样得到下一个字符，记作z1，decoder网络再拿z1作为输入，更新状态s2，并且输出预测的概率p2，根据p2做抽样，得到下一个字符z2，.......不断重复此过程，如果得到的字符是停止符，就停止此过程，返回生成的序列，z1,z2,.....。

注意力机制让任务处理系统找到与当前任务相关显著的输入信息，并按重要性进行处理，从而提高输出的质量。不需要监督信号，可推理多种不同模态数据之间的难以解释、隐蔽性强、复杂映射关系，对于先验认知少的问题，极为有效。解决长距离依赖问题 ，提升任务性能。

但是对RNN有注意力偏置问题，Coverage机制可以缓解注意力偏置问题。coverage向量表示过往时刻的所有注意力机制的累加，其告诉模型，输入中哪些部分是已经被注意力关注过的，哪些没有。为了使得模型不过多地关注已经关注过的区域，所以将coverge向量作为下一步的注意力的构成部分，这样下一步生成的注意力分布就会有意识地减少已经关注过的区域的概率。

具体设计如下：

Encoder采用BiGRU结构，BiGRU-Attention 模型共分为三部分：文本向量化输入层、 隐含层和输出层。其中，隐含层由 BiGRU 层、attention 层和 Dense 层（全连接层）三层构成。

decoder的结构为单向GRU。

由于训练样本中的句子长度不一致并且相差很多，所以引入了Padding Mask机制。由于文本一般是不定长的，所以在进行 batch训练之前，要先进行长度的统一，过长的句子可以通过truncating 截断到固定的长度，过短的句子可以通过 padding 增加到固定的长度，但是 padding 对应的字符只是为了统一长度，并没有实际的价值，因此希望在之后的计算中屏蔽它们，这时候就需要 Mask。



1. 数据处理部分

数据预处理代码，同时处理了这种影响训练的词。

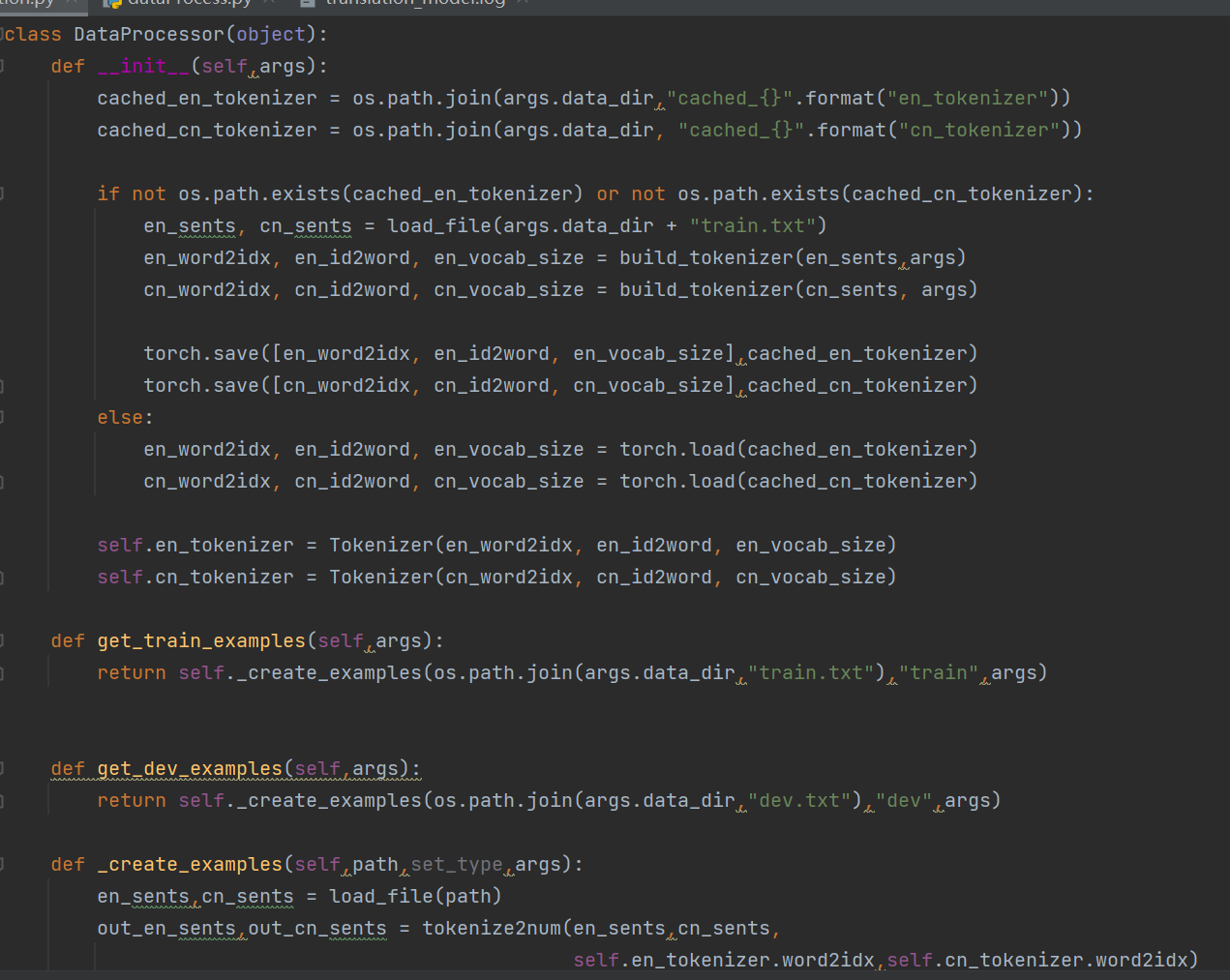




数据读取代码



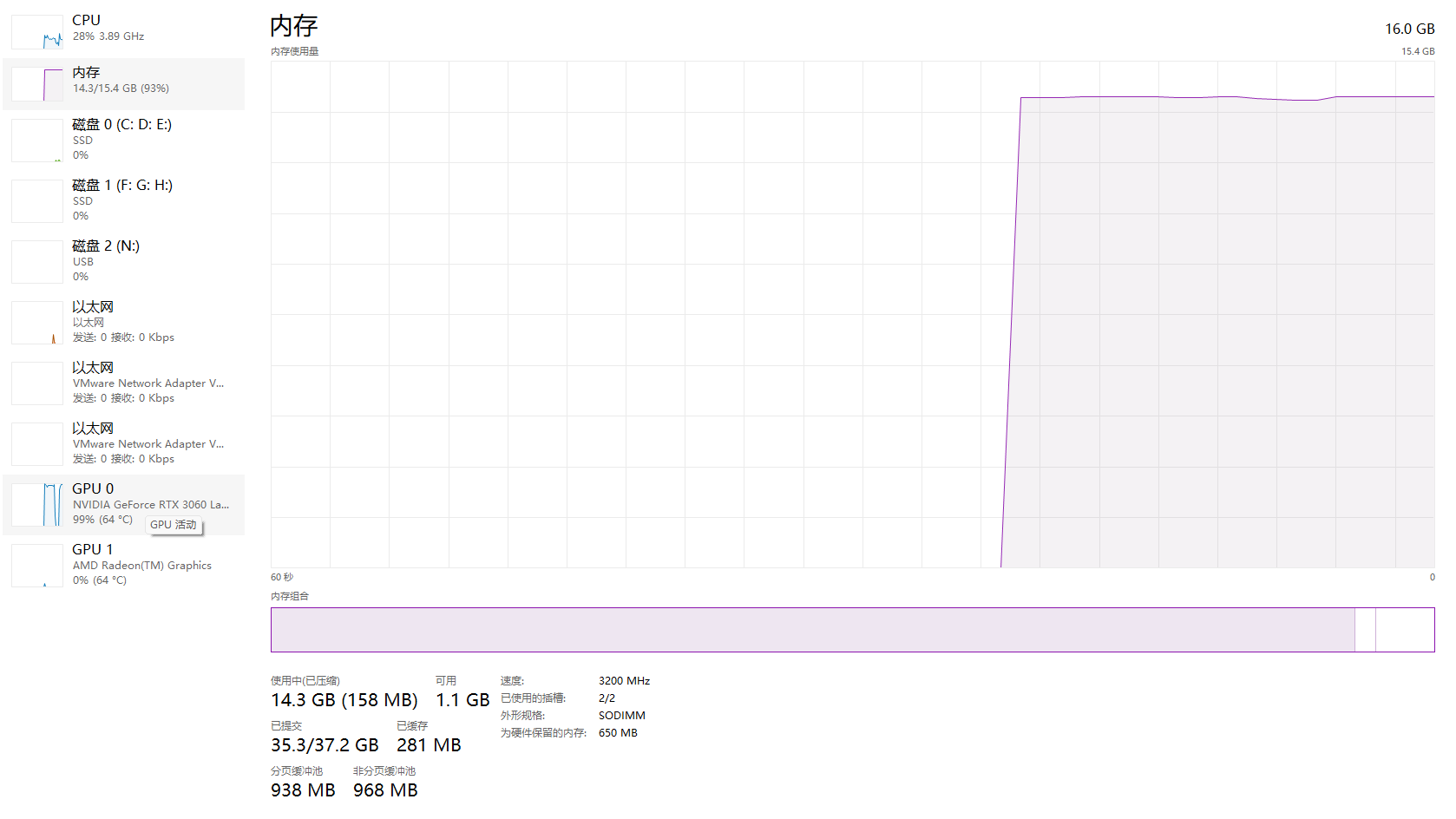
数据加工代码





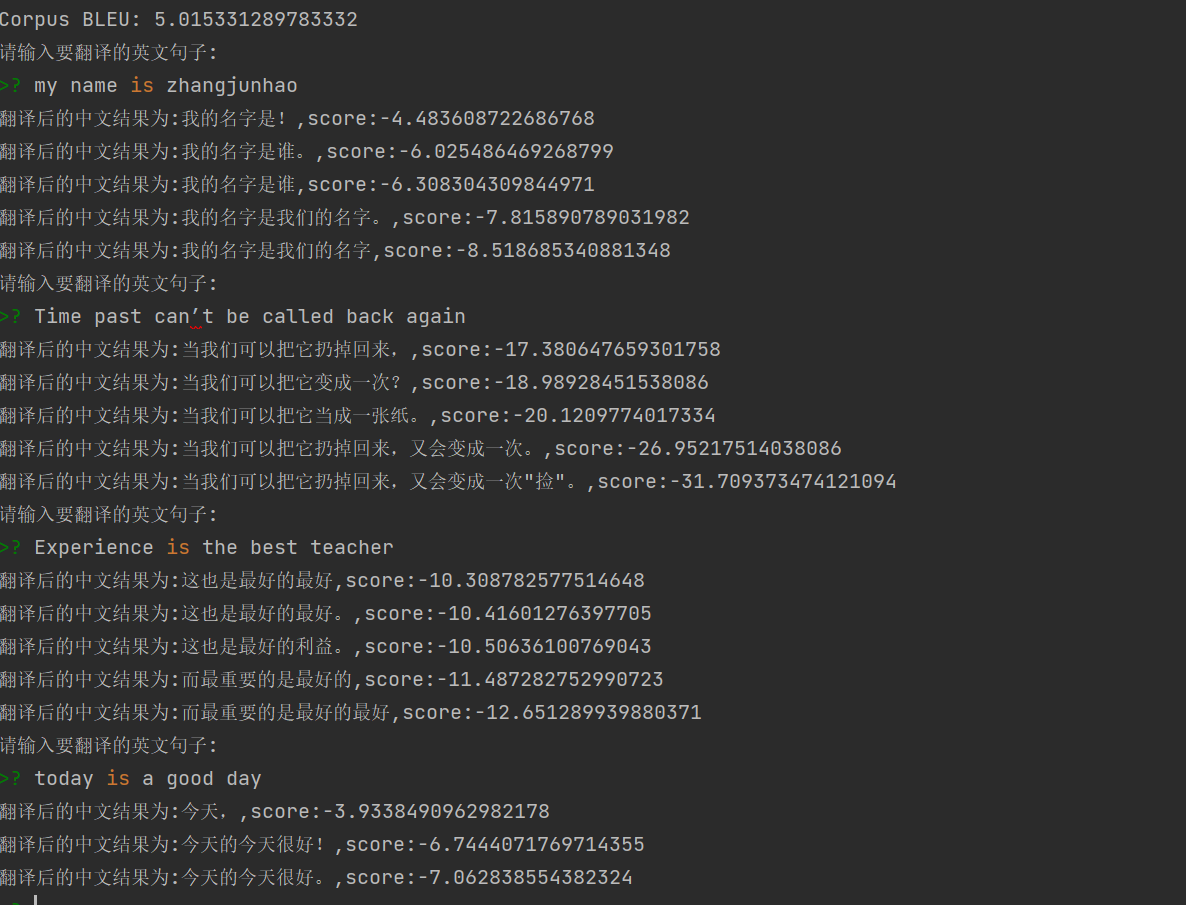
1. 训练过程

总共10个Epoch，每个Epoch 跑3万条数据，5万的时候电脑就会炸，3万条的时候内存也满了，显存也满了。



其他参数的设置，batch\_size=16，学习率在动态调整初始时设置的为0.0005,因为epoch较小所以使用了AdamW优化器。

训练结果：



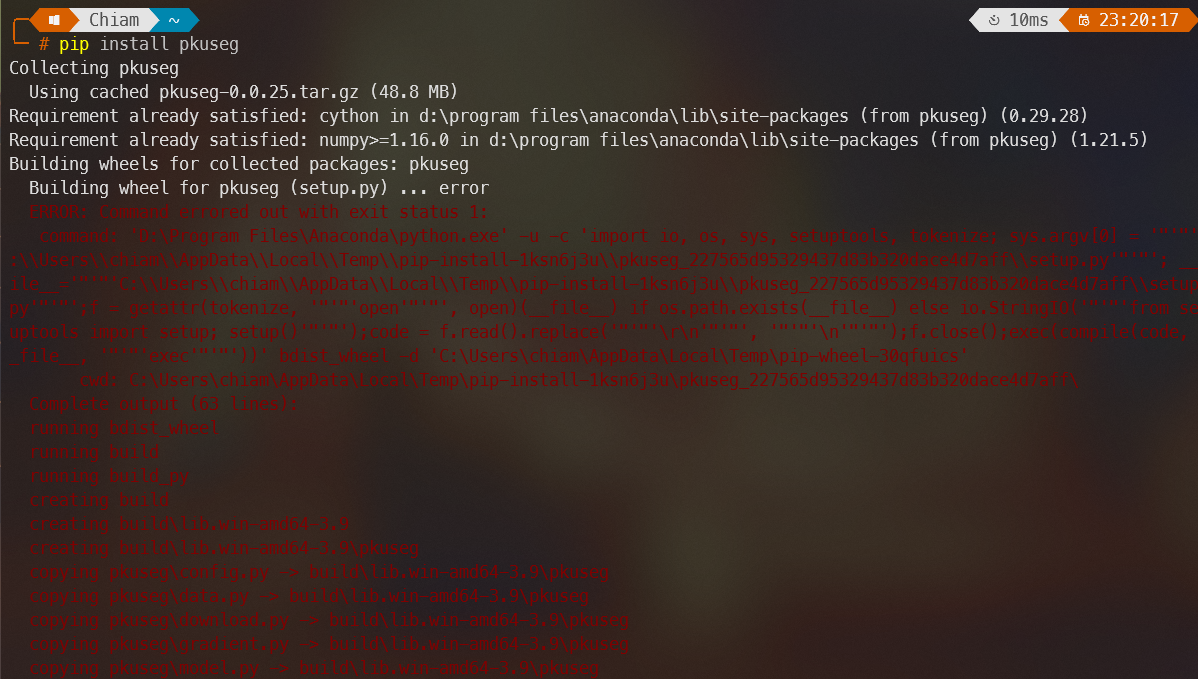
有点意思，但是又差很多，我觉得问题主要出现在以下原因，第一点是我的计算资源很有限，第二就是训练样本中有些奇怪的句子。





1. 实验中遇到的问题：

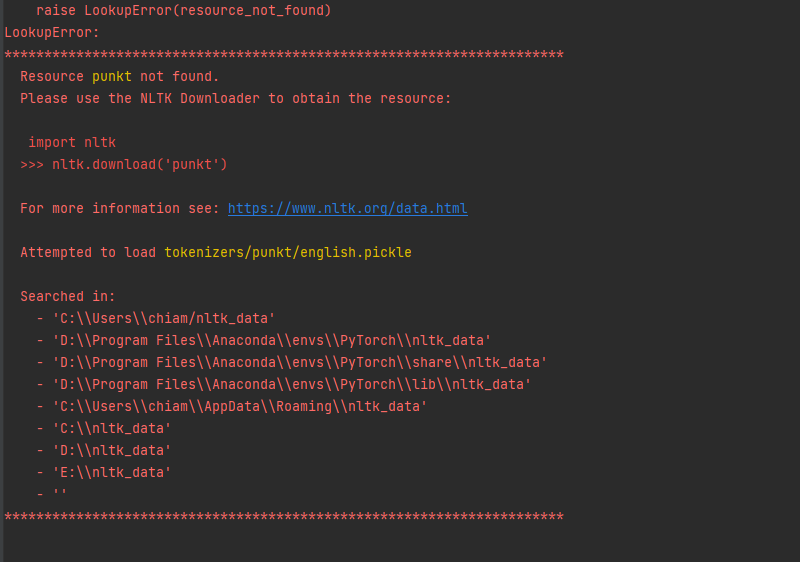
4.1.Pkuseg 安装问题



无论使用Conda还是Pip都无法安装usage，查了各种原因发现只支持3.8，由于我的Pytorch和cuda各种是之前配好的，所以不能整体换到3.9，所以采用源码进行编译，编译也报错，后来使用Github上的不封装模型的源码，就能编译成功，然后在手动导入模型，终于成功安装，耗时两天半。

4.2. nltk遇到的punkt问题

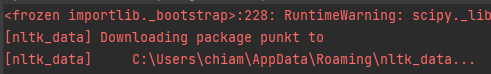
问题描述：



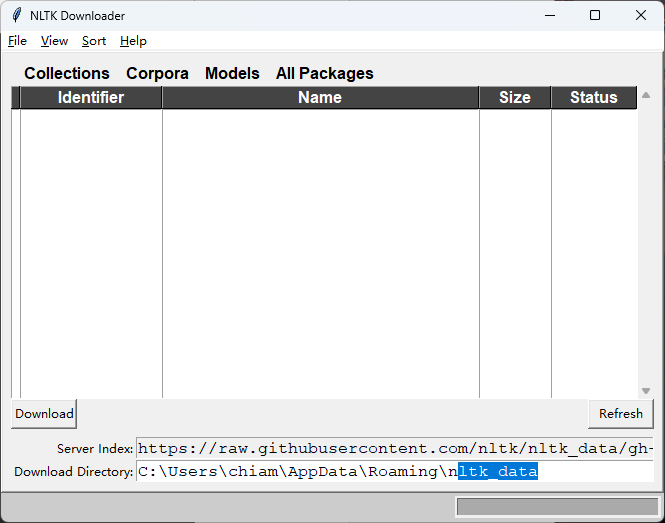
**解决方法1：**

**在python中输入代码即可完成下载：**

nltk.download('punkt')



然后会弹出GUI来，然而查不到所需要的东西，疯狂Refresh还是没用，使用Proxy之后仍然无法访问，遂采用第二种办法。

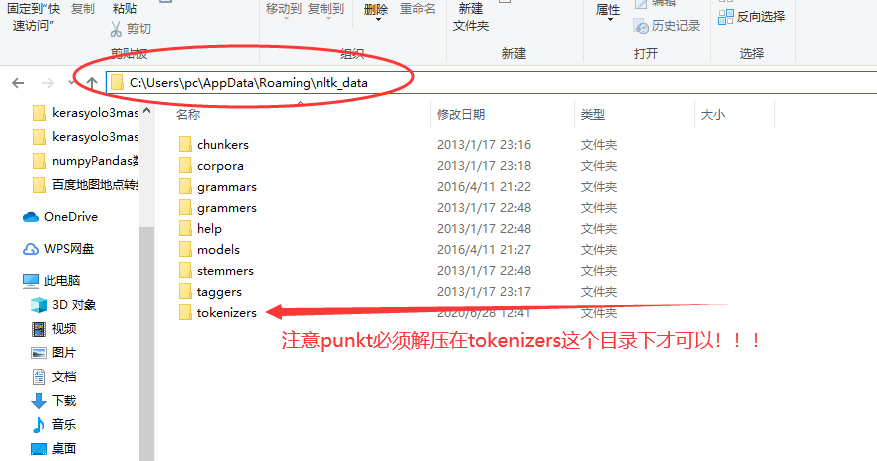


**解决方法2：**

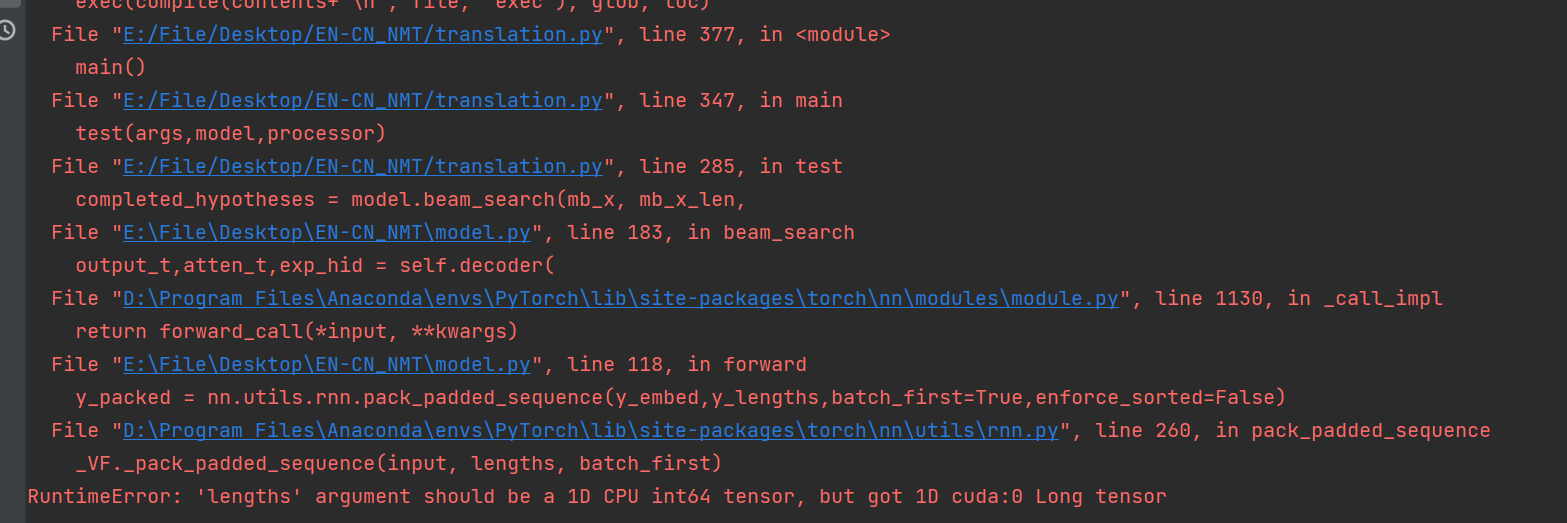
**下载punkt安装包**

**下载完成后解压在任一nltk\_data路径下的tokenizers目录下即可**

**这里是我的任意一个路径：**



4.3. RuntimeError: ‘lengths’ argument should be a 1D CPU int64 tensor, but got 1D cuda:0 Long tensor



报错内容

RuntimeError: ‘lengths’ argument should be a 1D CPU int64 tensor, but got 1D cuda:0 Long tensor

原因分析

因为我使用的是高版本的 torch 1.7.1，这个报错是由于torch 1.5 以上BiLSTM升级导致的。

解决方案

转到CPU运行代码就不报错了！

将报错的位置参数lengths转化为cpu 类型:

lengths.to("cpu")

1. 补充：

单模型240M了，所以模型就不放到提交的代码中，如果有测试需要，以下为完整版，解压即可运行，阿里云链接：https://www.aliyundrive.com/s/XWBX7gziRGq

1. [] Gehring J, Auli M, Grangier D, et al. Convolutional sequence to sequence learning[C]//International Conference on Machine Learning. PMLR, 2017: 1243-1252. [↑](#footnote-ref-1)
2. [] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate[J]. arXiv preprint arXiv:1409.0473, 2014.. [↑](#footnote-ref-2)
3. [] Vaswani et al. Attention is All You Need. NIPS 2017. [↑](#footnote-ref-3)