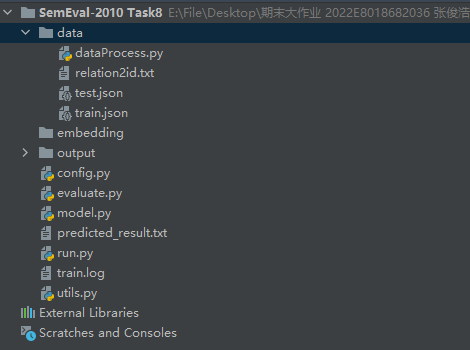
## SemEval-2010 Task8 关系抽取作业报告

文件目录



Data:存储预处理后的文件，数据预处理文件dataProcess.py也在此文件夹。

Embedding:存储预训练的glove.6B.100d数据

Output:用于存储输出的模型和训练过程中所使用的config,高手写的代码就是不一样，这是我第一次见这样加载训练参数。

Train.log:用于存储训练的输出

Run.py:主函数，训练和测试依次进行。

任务概述与理论支撑

为了构建知识图谱，从结构化(如表格)、半结构化(如JSON)和非结构化(如纯文本)数据中获取形式为(事物1，关系，事物2)的三元组的过程称为关系抽取(relation extraction)。一般情况下，我们会尽量把关系抽取抽象成若干三元组的抽取，而不会做n元组(n>3)的抽取。在NLP中，实体关系抽取则是致力于从自然语言文本中识别出实体对并判断实体间特定语义关系的任务，输入的是一句文本，输出的是spo三元组(subject-predicate/relation-object)。

比如：

比尔盖茨是微软的CEO：CEO(比尔盖茨, 微软)

信工所在闵庄路：Located-in (信工所, 闵庄路)

关系类型

a：正常关系

b：关系重叠：一对多

“张学友演唱过《吻别》《在你身边》”中存在2种关系：（张学友-歌手-吻别）、（张学友-歌手-在你身边）

c：关系重叠：一对实体间存在多种关系

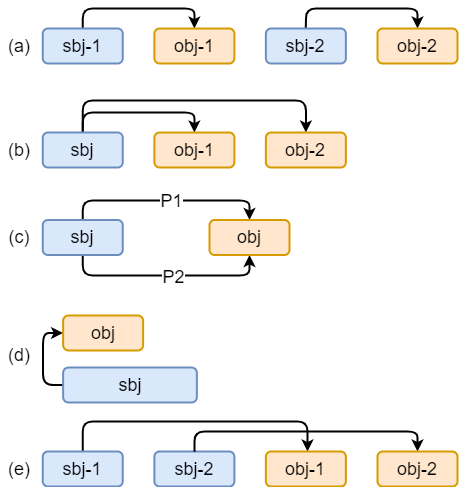
“周杰伦作曲并演唱《七里香》”中存在2种关系：（周杰伦-歌手-七里香）、（周杰伦-作曲-七里香）

d：由实体重叠导致的复杂关系

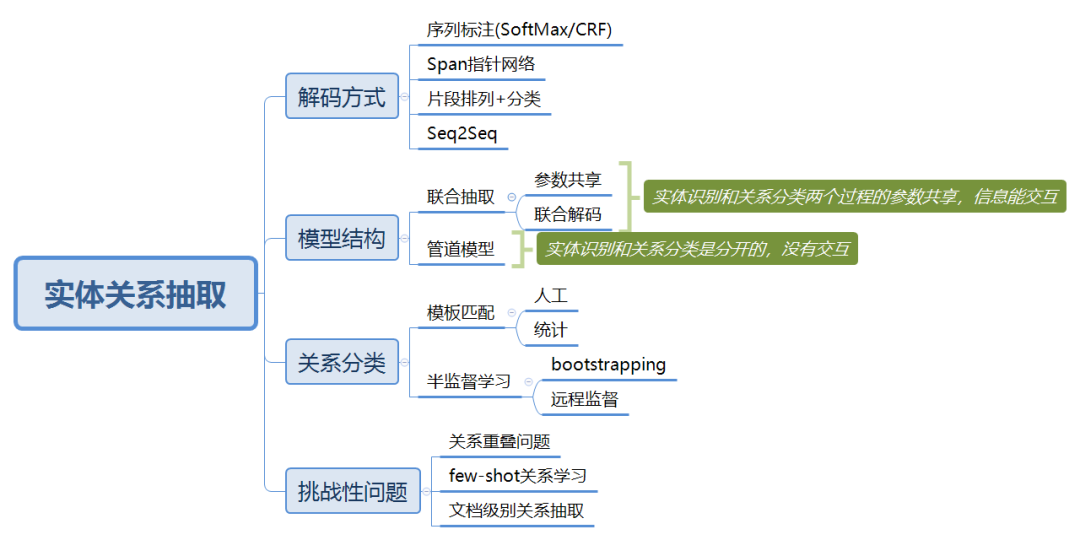
“《叶圣陶散文选集》”中存在关系：（叶圣陶-作品-叶圣陶散文选集）

e：由关系交叉导致的复杂关系

“张学友、周杰伦分别演唱过《吻别》《七里香》”中存在关系：（张学友-歌手-吻别）、（周杰伦-歌手-七里香）



实体关系抽取



数据集及数据处理

Trial Dataset：试验数据集于2009年8月30日发布，它包含前五个关系的数据。但是，其中也包含了一些其他四种关系的引用，这些数据在试验数据集上可以被视为Other关系，而不必多加处理。Training Dataset:训练集包含8000个样例，涵盖上文提到的9+1中关系。Development Dataset：没有提供官方开发集，但是参与者可以使用该部分训练数据集来调整期参数，如使用交叉验证。Test Dataset：测试集包含2717个样例，涵盖上文提到的9+1中关系，于2010年3月18日发布。

WordNet senses提示：和SemEval-2007 Task 4不同，此处不提供人工标注的WordNet senses，会使得任务更加真实。

|  |  |  |
| --- | --- | --- |
| 类别 | 训练集中的数目(比例) | 测试集中的数目(比例) |
| Other | 1410 (17.63%) | 454 (16.71%) |
| Cause-Effect | 1003 (12.54%) | 328 (12.07%) |
| Component-Whole | 941 (11.76%) | 312 (11.48%) |
| Entity-Destination | 845 (10.56%) | 292 (10.75%) |
| Product-Producer | 717 ( 8.96%) | 261 ( 9.61%) |
| Entity-Origin | 716 ( 8.95%) | 258 ( 9.50%) |
| Member-Collection | 690 ( 8.63%) | 233 ( 8.58%) |
| Message-Topic | 634 ( 7.92%) | 231 ( 8.50%) |
| Content-Container | 540 ( 6.75%) | 192 ( 7.07%) |
| Instrument-Agency | 504 ( 6.30%) | 156 ( 5.74%) |

原始数据格式

8001 " The most common <e1>audits</e1> were about <e2>waste</e2> and recycling . "

Message-Topic(e1 ,e2)

Comment: Assuming an audit = an audit document

经过处理之后的数据格式：

{"id": "8001", "relation": "Message-Topic(e1,e2)", "sentence": ["The", "most", "common", "<e1>", "audits", "</e1>", "were", "about", "<e2>", "waste", "</e2>", "and", "recycling", "."], "comment": " Assuming an audit = an audit document."}

在该模型词嵌入word\_embedding中使用GloVe算法，，GloVe的全称叫Global Vectors for Word Representation，它是一个基于全局词频统计（count-based & overall statistics）的词表征（word representation）工具，它可以把一个单词表达成一个由实数组成的向量，这些向量捕捉到了单词之间一些语义特性，比如相似性（similarity）、类比性（analogy）等。我们通过对向量的运算，比如欧几里得距离或者cosine相似度，可以计算出两个单词之间的语义相似性。glove.6B.100d.txt是该算法需要用的预训练数据，100d为100维的数据。

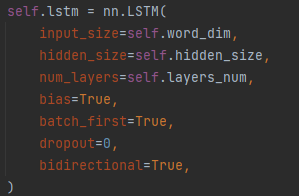
模型结构

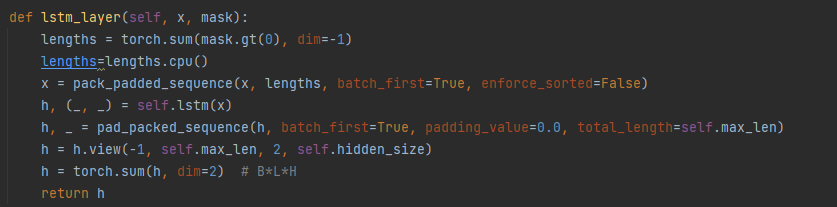
输入层（Input layer）：将句子输入模型

嵌入层（Embedding layer）：通过训练，将每个词都映射到一个较短的词向量上来所有的这些词向量就构成了向量空间，进而可以用普通的统计学的方法来研究词与词之间的关系。

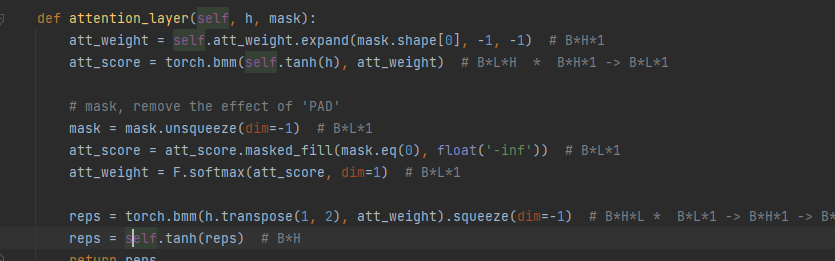
LSTM层（LSTM layer）：利用BiLSTM从词向量中获得特征

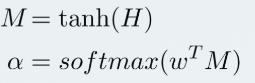
Attention层（Attention layer）：生成权重向量，将每个时间步长的单词级特征与权重向量相乘，合并成句子级特征向量





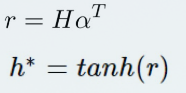
输出层（Output layer）： 使用句子级特征向量进行关系分类





这两公式由这两步实现:

att\_weight = self.att\_weight.expand(mask.shape[0], -1, -1) att\_score = torch.bmm(self.tanh(h), att\_weight)



这两公式由这两步实现:

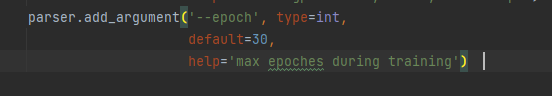
reps = torch.bmm(h.transpose(1, 2), att\_weight).squeeze(dim=-1) reps = self.tanh(reps)

此时reps 就是用于最终分类的hidden state

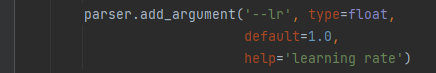
训练过程

训练的时的参数设置，这是我第一次见别人这样设置参数，感觉之前有点孤陋寡闻的，他这样感觉模型的复用性更强，config是单独一个py文件，运行时加载这样就不用改主程序里代码，但是到我手里还是变得乱糟糟的。

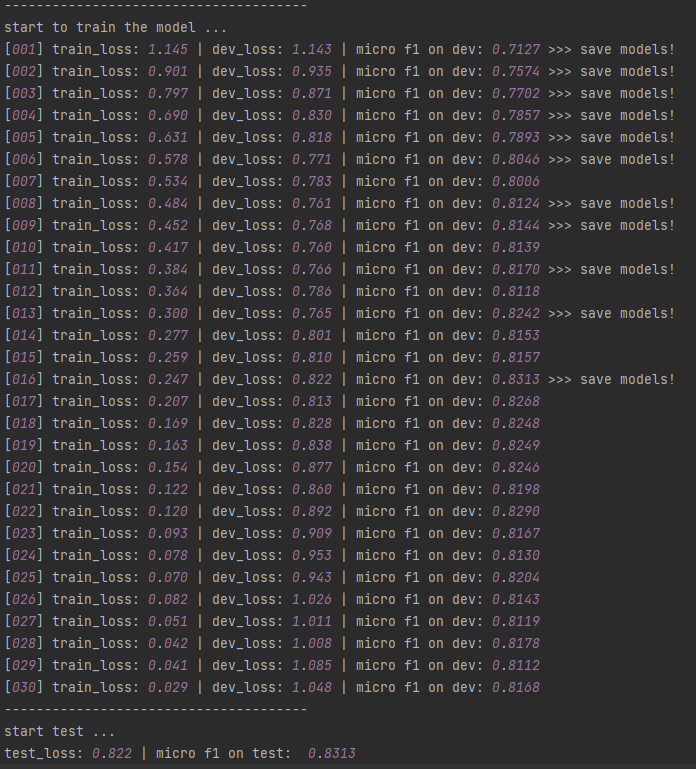
总共跑了30个epoch:



学习率初始设置为1.0



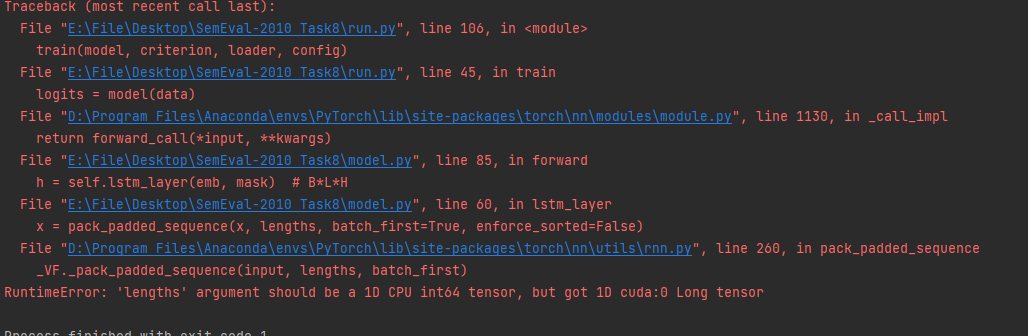
每个batch的大小设置为10，除了线性层丢弃0.5的神经元其余都为0.3用于防止过拟合。(Bi)LSTM层中隐藏单元的维数设置为100。句子最大长度设置为100。训练结果如下：



Perl工具在我做上次作业的时候就配置不了，所以还是使用Python代码进行计算。经过测试F1指数为0.8313。

遇到的问题

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

跟昨天一样的内容，因为我使用的是高版本的 torch，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")

补充：

glove.6B.100d和模型架起了480M了，所以模型和该文件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/ASoh5A365dN