《作业3 信息抽取系统》  
实验报告



作业3 信息抽取系统

1. 概述
   1. 数据集介绍

数据集来源于Kaggle上的竞赛[NBME - Score Clinical Patient Notes](https://www.kaggle.com/competitions/nbme-score-clinical-patient-notes?rvi=1)。数据是美国医学执照考试(Medical Licensing Exams)中的病例考题。竞赛的目标是提取病历中的关键词，并且在训练集中，关键词已经给出位置。这意味着抽取数据时，不需要“凭空”生成出实际的关键词，而只需要找到关键词的位置即可。

数据集分为三个文件：features.csv, train.csv, patient\_notes.csv。

Feature内部结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 说明 | 样例 | 备注 |
| feature\_num | 特征id | 000 |  |
| case\_num | case id | 0 | 没有用到 |
| feature\_text | 特征文本 | Family-history-of-MI-OR-Family-history-of-myocardial-infarction |  |

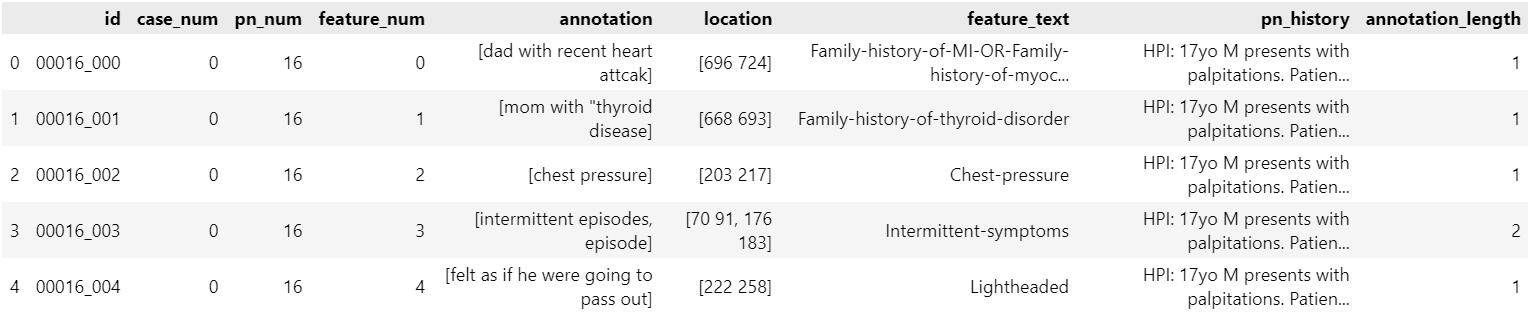
Train.csv

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 说明 | 样例 | 备注 |
| id | 训练数据id | 00016\_000 |  |
| case\_num | case id | 0 | 没有用到 |
| pn\_num | 病人记录id | 00016 |  |
| feature\_num | 特征id | 000 |  |
| annotation | 标记的实际文本 | ['dad with recent heart attcak'] | 可能为空，或者多个 |
| location | 标记文本在原文的位置 | ['696 724'] | 可能为空，或者多个 |

patient\_notes.csv。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 说明 | 样例（局部内容） | 备注 |
| pn\_num | 病人记录id | 00000 |  |
| case\_num | case id | 0 | 没有用到 |
| pn\_history | 病人的病例 | 17-year-old male, has come to the student health clinic complaining of heart pounding. Mr. Cleveland... |  |

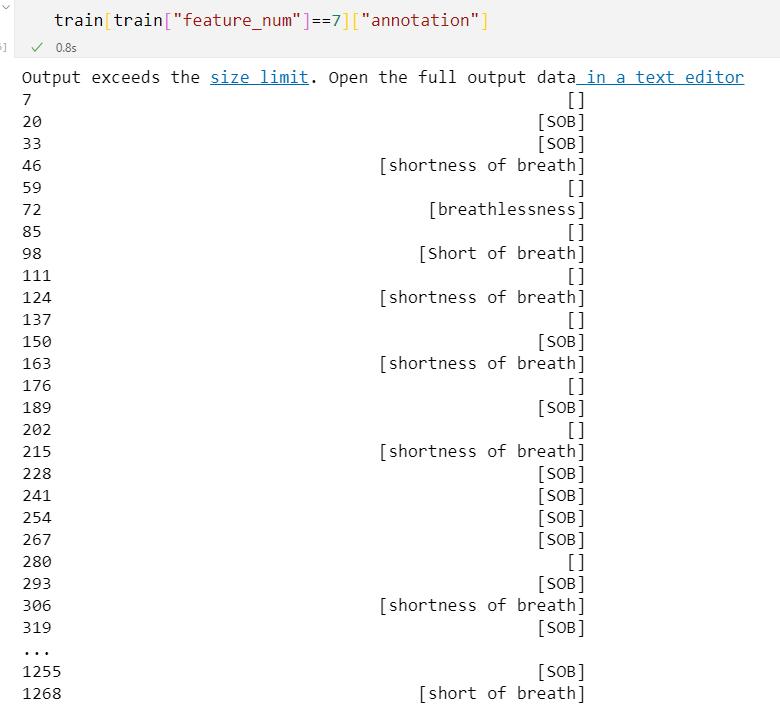
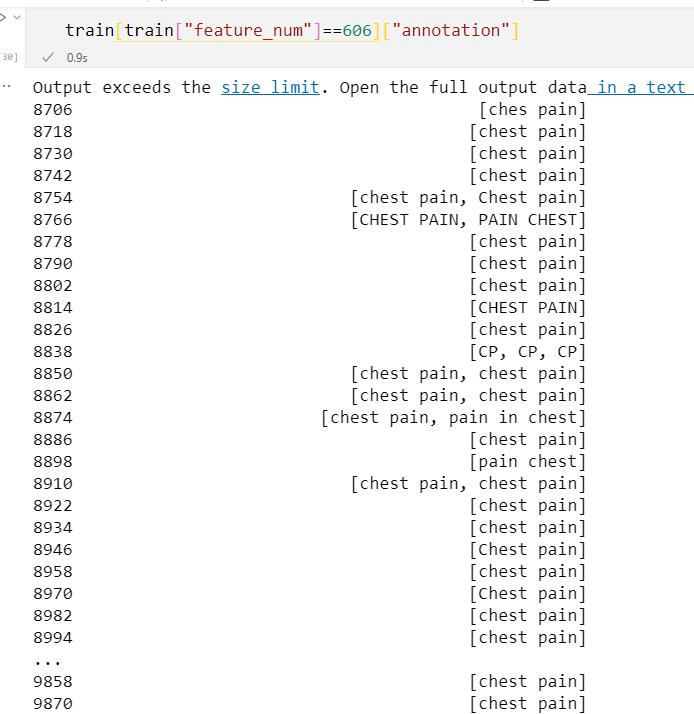
为了方便训练，通过id所指示的外键将三个表连接在一起：



* 1. Feature的解释

这里feature可以看成是一个类别。例如，feature=606的feature text是chest pain，表示胸痛。在真实病例（左图）里面，chest pain有时会写错了，比如少写一个t，或者写成简写CP，又或者写成 pain in chest.

又比如feature=7是shortness of breath，即呼吸困难。在实际病例里面（右图），大部分被简写成SOB，有时候有语法错误，写成short of breath，或者直接造了一个词，写作breathlessness。



可以注意到有时候是提取不出关键词的，比如feature=7的shortness of breath中就有不少没有标注的数据。我们不需要对它做特殊处理，因为返回空的列表也是一个正常的返回。

* 1. 信息点

我们要抽取的信息点正是features。其中一共有143种feature，每个病人的病例可能有多个feature。下面是全部的features：



1. 系统结构和算法

2.1 序列预测

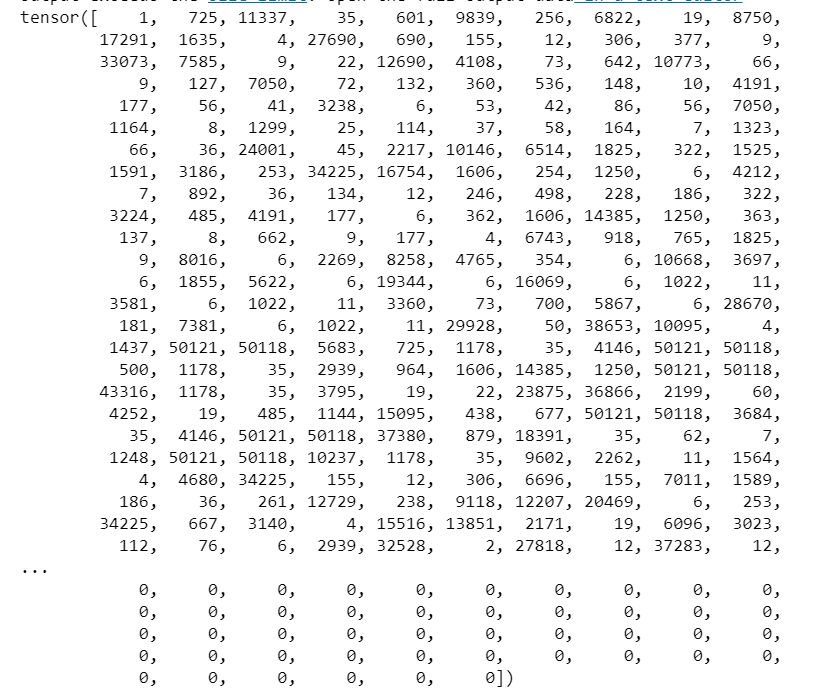
这个问题比较类似NLP中的序列标注问题。我们的目标是读取文本和特征文本，返回可能的特征所在的位置。它采用的是BIO的标签风格。所谓BIO的标签，即Begin，Inside, Outside。其中Begin表示标签开始的地方，Inside表示标签内部或者结束的地方，其他均为Outside标签。因为一次只需要预测一个种类，因此只需要这三类标签（如果一次需要预测多个种类，则需要多个B标签和I标签）

对于文本序列预测问题，比较常用的方法都是用预训练模型进行微调。我们选用预训练模型是DeBERTa。

DeBERTa 是微软在Bert的基础上改进的一个预训练模型，它主要在注意力机制、掩码器方法做了调整，同时使用的是一种虚拟对抗的训练方法，相比于Bert有显著的提高。

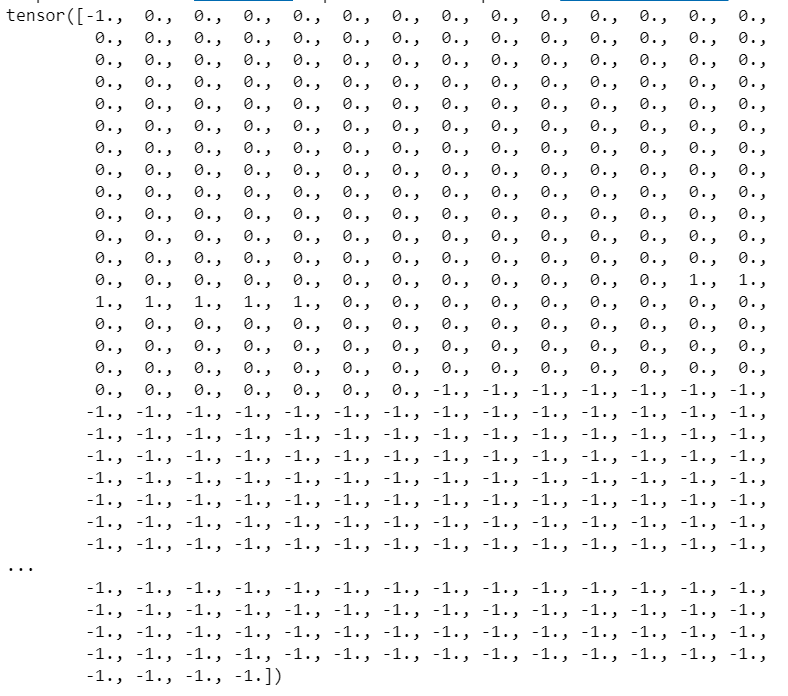
下面重点介绍一下Fine tune部分。Fine tune指的是在预训练模型的基础上，加入我们需要预测的输入和输出，对模型内部参数进行微调。

我们的输入是一个词向量。这里的词向量不是one-hot向量，而是简单地编码后，类似数组的一个向量。例如下面是一段病例的文本：PI: 17yo M presents with palpitations. Patient reports 3-4 months of intermittent episodes.....，我们将其传入tokenizer函数，一个用于把半词或者词组转化成token（数字）的函数，可以得到：



其中，1表示语段的开始，725是一个特殊字符，11337是”PI”,35是冒号，601是17.. 以此类推。这样我们就将语段转化成了向量。因为pytorch要求输入长度是固定的，因此我们将其设定为512，如果长度没有达到512，会在后面填0。

对于输出，我们期望得到的是它们的位置。但是类Encoder模型对每个输入都有一个输出。比较常见的一种输出方式是每个输入，输出要么是越界，要么是无关项，要么是正确的标签。我们使用python transformer内置tokenizer，将之前annotation位置重新进行序列化，可以得到类似下面的结果：



上图中，0的部分表示的是无关项，-1的部分是attension\_mask为0的地方，1的地方是我们的标签。

有了输入和输出，下面编写数据集，输入模型进行训练即可。

训练的模型是在DeBERTa输出的外面套上一层Dropout和全连接层，增加非线性。

对于模型的损失函数，我们使用的是常用的交叉熵损失函数，但是在外面多套了一层lostics函数，这样会比普通交叉熵更加稳定一些。我们直接使用的是pytorch的BCEWithLogitsLoss。

在模型的评估上面，我们主要是对序列覆盖做F1分数。

例如，实际的位置是20,25，预测的结果是18,24。那么，我们建立两个等长的向量

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 实际 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 预测 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

这样，我们就可以做出TFPN的表格：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 预测-1 | 预测-0 | 总数 |
| 真实-1 | 5 | 1 | 6 |
| 真实-0 | 2 | 0 | 2 |
| 总数 | 7 | 1 | 8 |

进而可以计算准确率为5/6,召回率为5/7.，F1分数为0.76

如果实际值为空列表，那么跳过这个预测。

这个比赛没有提供测试集，因此只能在训练集里划出一部分作为测试集。我们将训练集的20%作为测试集，在Kaggle平台上在线训练了5个Epoch。然后将参数保存为pth文件，下载到本地读取使用。

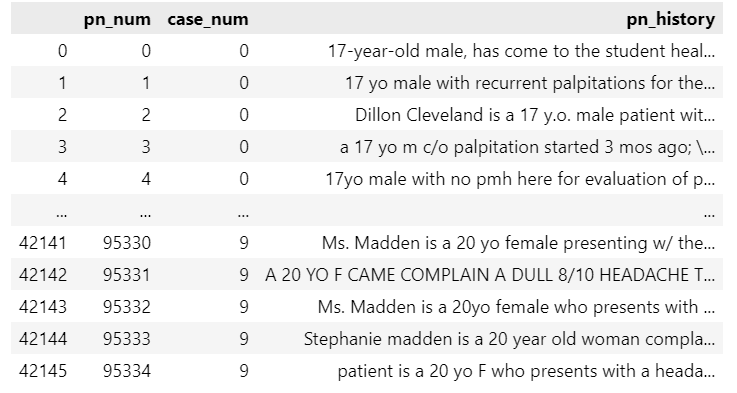
最后训练结束后，平均损失为0.0072。对测试集进行预测，预测的F1分数为0.86。这是一个相对比较高的值，说明训练的效果还不错。

我们划分了5个部分，并做了5个epoch的训练。每个epoch选一个部分做测试集，剩下四个部分做训练集。5个epoch测试集得到F1分数如下：

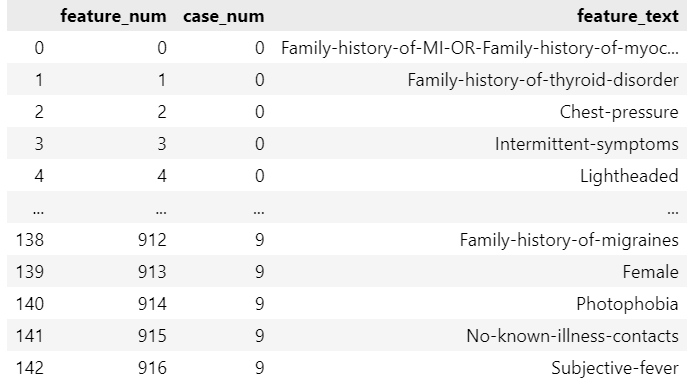
|  |  |
| --- | --- |
| 序号 | F1分数 |
| 1 | 0.8319 |
| 2 | 0.8569 |
| 3 | 0.8659 |
| 4 | 0.8613 |
| 5 | 0.8648 |

2.2 数据的扩充

在模型上进行改进对我们来说还是比较困难的。在比赛给定的数据集中(patient\_notes)，有一些病历是没有出现在训练集里的。



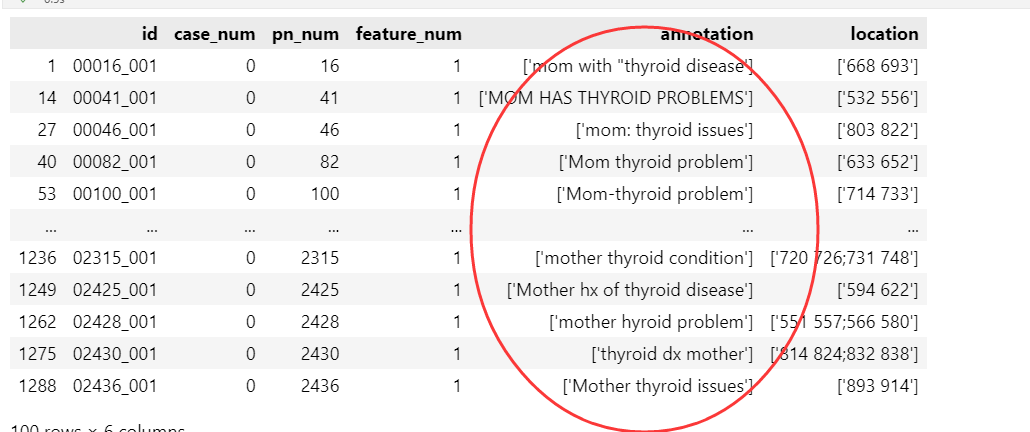
但是，它提供了一个case\_num。而feature里面也有一个case\_num，这是之前没有用到过的。



于是，我们可以根据case\_num的feature，去对应的pn\_num里面查找，看是否能找到feature。

直接取搜索的话数据量太大了，而且会与原来的数据重复。我们的思路大致是这样的:

首先筛选出没有进入train的pn\_num。然后遍历这个列表。根据每个note对应的case，我们去找到对应的features。features的文本直接匹配是匹配不到的，因此还要从原来的训练集里面，提取出对应feature的annotation集合。然后取集合里的字符串，去patient\_notes里面匹配。



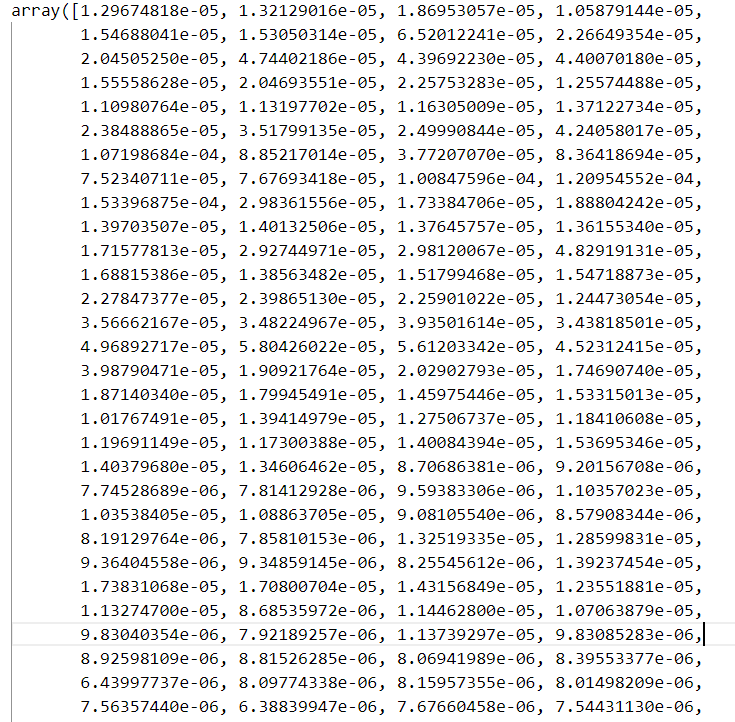
我们最终得到了876个新的训练数据，而原来有14300条训练数据，增长了大约6%的训练数据。

我们将新数据划分fold后训练，最后训练结果略好于原来的结果：

|  |  |  |
| --- | --- | --- |
| 序号 | 导入新数据 | 原数据 |
| 1 | 0.8258 | 0.8319 |
| 2 | 0.8615 | 0.8569 |
| 3 | 0.8657 | 0.8659 |
| 4 | 0.8697 | 0.8613 |
| 5 | 0.8701 | 0.8648 |

2.3 展示程序的设计

和上一个作业类似，我们使用了web平台来进行展示。后端使用的是python的Fastapi框架。后端将model读取到内存和GPU中，然后等待请求。一个请求会包含同一文本的多个特征。当请求到来的时候，首先将请求转化为上文所说的序列向量，然后把请求内的多个特征合并成一个张量Tensor，随后送入模型进行预测。预测得到的结果是一个上文所述的向量：



这里这个向量长度还不等于我们原文的向量（可能一个词组是一个token）。我们需要将其展开并映射回原来的句子上。

最后我们就可以找哪些值可能是标签了。我们设定一个阈值，比如说0.5，提取所有阈值大于0.5的区间，这样就可以得到预测的结果了。

可能会得到多个可能相互覆盖的区间，因此在传给前端之前还需要做一个预处理，把区间进行合并。具体的算法如下：

Intervals = 区间列表

Results = 结果列表

Result.append(interval[0])

For interval in Intervals[1:]:

If Result.last().right() >= interval.left(){

Result.last.right().set(max(interval.right(),Result.last.right()))

}else{

Result.append(interval)

}

后端的API如下：

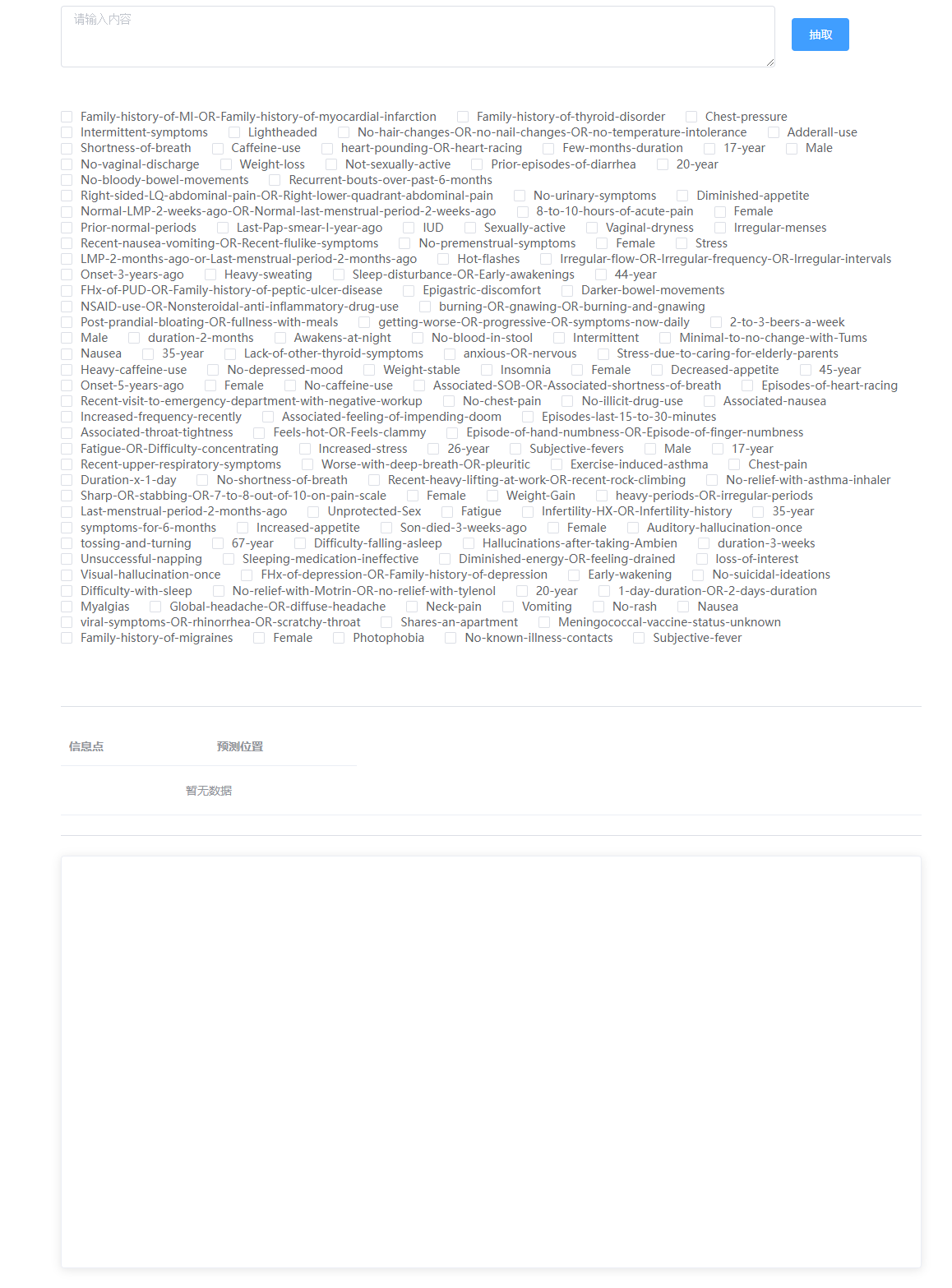
POST /query

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| text | str | 文本 |
| feature\_ids | list[int] | 需要预测的feature |

返回值：

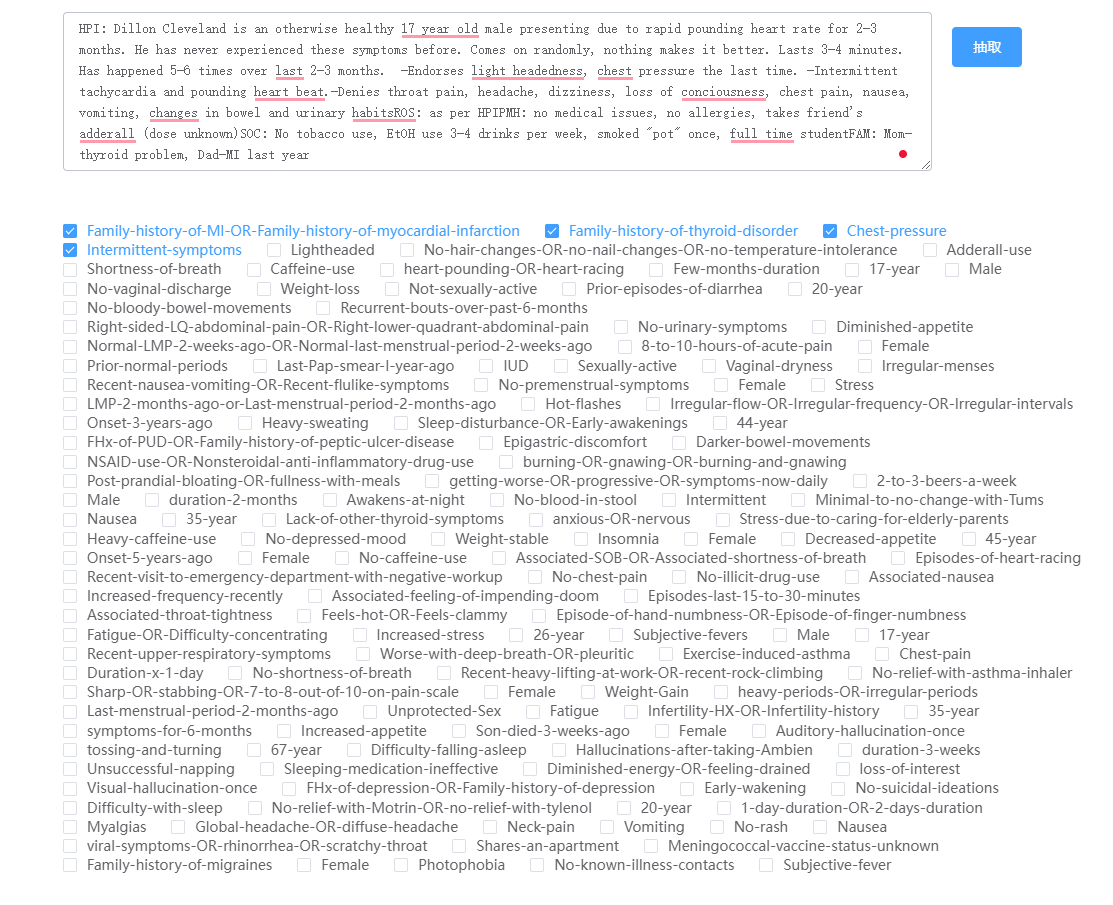
|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | | 说明 | 类型 |
| result(列表) | feature | 信息点名称 | str |
| pos | 预测的位置 | list[list[int]] |
| intervals | | 需要高亮的位置 | list[list[int]] |

前端的页面：

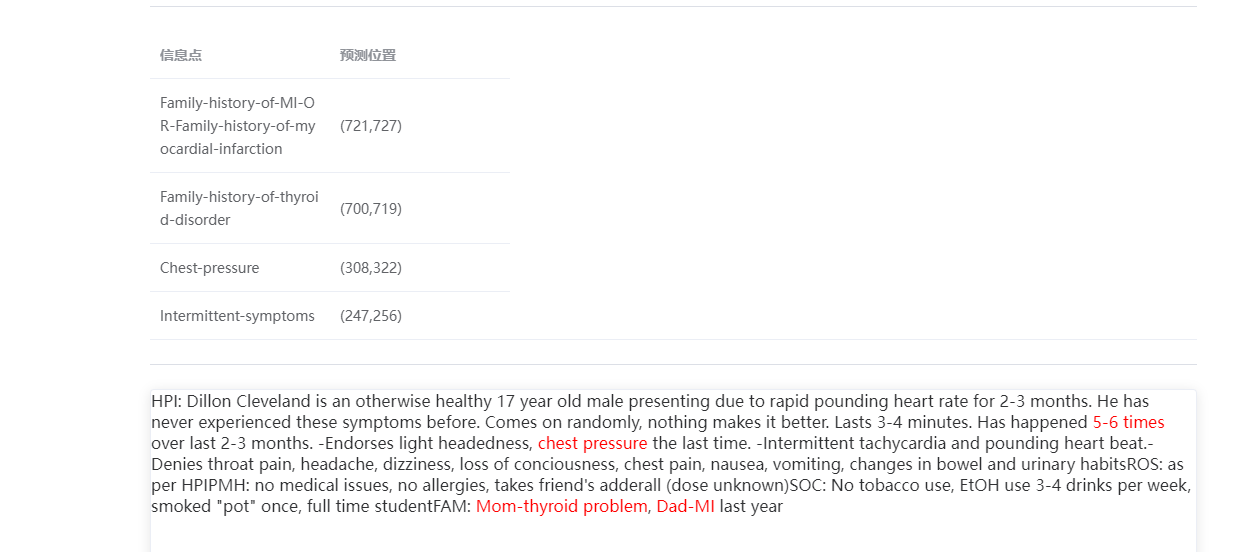


1. 成果演示

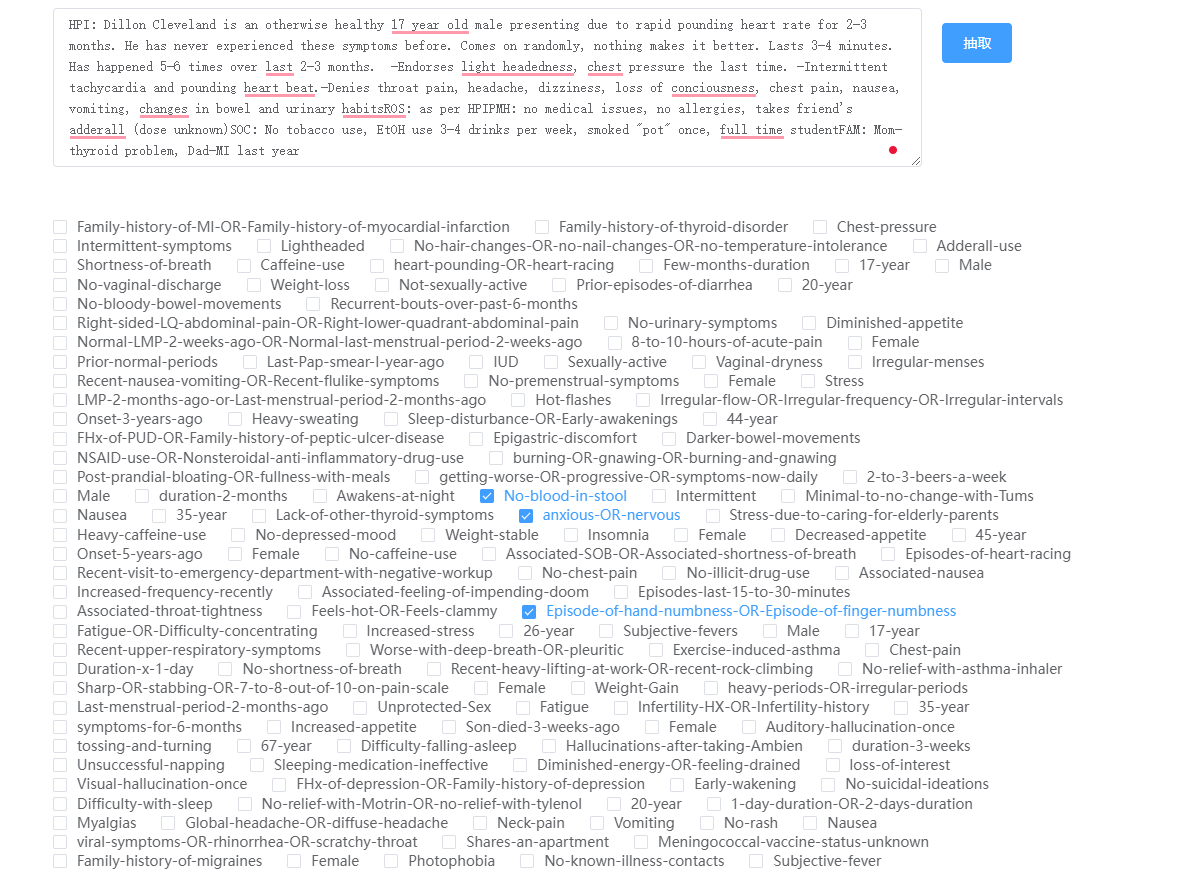
这里以输入病例” HPI: Dillon Cleveland is an otherwise healthy 17 year old male presenting due to rapid pounding heart rate for 2-3 months. He has never experienced these symptoms before. Comes on randomly, nothing makes it better. Lasts 3-4 minutes. Has happened 5-6 times over last 2-3 months. -Endorses light headedness, chest pressure the last time. -Intermittent tachycardia and pounding heart beat.-Denies throat pain, headache, dizziness, loss of conciousness, chest pain, nausea, vomiting, changes in bowel and urinary habitsROS: as per HPIPMH: no medical issues, no allergies, takes friend's adderall (dose unknown)SOC: No tobacco use, EtOH use 3-4 drinks per week, smoked "pot" once, full time studentFAM: Mom-thyroid problem, Dad-MI last year”为例进行演示。



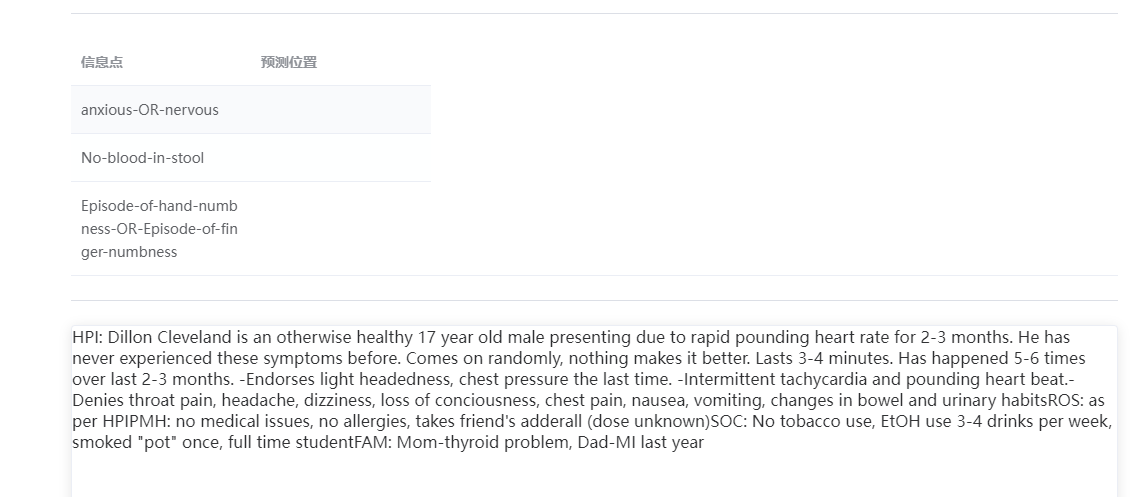
抽取的结果：



如果抽取的是不存在的信息点：



预测的位置为空



1. 总结

本次实验，我们使用深度学习预训练模型，进行了序列预测，提取了病例了病历中144种不同的信息点。通过这个实验，我们了解了预训练模型的简单使用和部署。