## **1.1重写了规则：**

1.LLM api的实验：

再次查阅了deepseek的开发者文档，发现原调用api多轮对话的格式有问题。本次完全遵循ds开发者文档的内容重写调用api的函数，并减少提示中的数据集和要求返回的数据集，原提示数据集为[0,800],待处理句子为[1000,1199].修改一批次为50行，八轮多轮对话作为历史微调api，要求返回50行，一共要求返回8次，400行作为测试。返回格式使用txt存储，不再执行“finishment默认为1,成功就赋值为0,失败就+1，当finishment<=3时组成map再次调用api返回结果，再匹配不上则结果返回None”的逻辑，只返回一次，检查ds在第一次返回中的正确率。

在之前的代码中，由于要求输出过长无法使用json格式。因为可能出现截断文本。之前的批次设定为一批次两百行，LLM的输出很有可能是不完整的,使用Json则必须引入额外的正则表达式来处理结果。

考虑到成本问题，无法使用json完成本次实验。精度问题考虑以后使用投票进行解决。

考虑deepseek可能更多使用中文语料库，英语地区AI可能在英语数据集上的表现会更加好，引入chatGPT进行对比，调用Azure AI中的满血模型4o（非推理模型）和o3-mini（推理模型）。

2: LLM微调实验

本次使用Deepseek-r1:1.5b的蒸馏模型进行微调。微调分为两种:全量微调和LoRA微调。LoRA微调使用的为unsloth框架，但是要求cuda 12.8，而hpc集群环境最高支持cuda12.2并且运行会覆盖当前环境导致一系列错误，所以暂时无法进行，需要全量微调实验结束后寻求其他的框架进行替代方案，在2080ti上进行了训练但是遇到了最后0.1epoch卡死的情况，经过排查是数据分片不对齐造成batch大小不一致，某些进程提前退出，其他进程等待之前进程完成导致的。由于训练的loss都还比较大，不具备评估的价值。而且LLM评估的成本非常高，所以采用训练损失和最后采用微调后的模型运行200行作为测试的算法。

LoRA微调的结果：

2.1 LoRA微调

查找前人的方法中，我见了一种说法使用epoch为2，但在接下来的训练中发现完全不可行，模型完全没有充分学习到数据特征，并且损失并没有完全收敛。在本阶段实验中epoch全部为2,但在下阶段实验中epoch将改为10.

设备为显卡2080Ti

设备为显卡v100\*4

训练交叉熵损失

修正数据集，首先将起始位置改成句子中的起始位置。其次使用新的评估指标。本次要求准确识别出句子中的实体，并需要给出位置和标签，计算准确率和召回率。使用10000行作为训练集，其余所有行作为测试集，要求测算出准确的实体、位置和标签，结果正确则返回df。注意这个df中的实体是模型预测正确和标签正确的实体以及位置和标签。采用多个NER模型中预测正确的部分取交集，作为可信数据集。这个数据集可以认为和训练集的风格较为一致，或者说更加贴近标签的描述。由于LLM Few-shot需要比较可靠的数据集，所以采用这个办法筛选。

现在重写精确度计算和召回度算法。并更新新的数据集，本次不再强制使用IOB标签的模型和不使用IOB格式的模型使用同一个数据集，而是分开进行评估。评估代码现在要计算三个指标：精确度，召回率和f1-score,分为三种级别，句子级别，实体级别和token级别。

实体级别以预测准确实体位置和标签为一次正确预测。token级则计算预测正确的token（一个实体可能包含多个token，一个句子可能包含多个或者零个实体）。

三种级别都需要计算三种指标。如果句子级别预测正确，则添加进True\_df.csv中，作为可信数据集.：

由于采用这种方法spacy的三个预训练管道在九个数据的指标上都取得了极其糟糕的成果，猜测模型很难学习实体位置的标注。因此再添加去除序列预测的指标作为参照，也就是一共14个指标。

因为在实际的应用中坐标虽然可以作为确定实体的唯一键值，但是要求一个语义理解的模型去预测位置显然是不合理的，后续追加精度的算法可能会追加检测实体的坐标然后根据实体和坐标去预测对应的标签。

在改进后的算法中,spacy的管道达到了要求的精度，由于训练集和测试集完全分开，可以认为结果真实有效。

追加重要更改：不再要求模型返回预测实体的位置。同时改写算法。新的算法是基于实体级，当模型实体与验证集实体不一致时，计入召回率，当模型实体与验证集一致但标签不一致时。算法将直接与对应句子比较来确定是第几个token。

纯左对齐问题

传统的交叉熵损失会在 Token 级别进行严格对齐；如果模型在中途多生成/少生成一个符号，就会导致其后续所有 Token 的对齐错位，进而造成整段后续输出都被视为错误。这种情况对“实体抽取”一类任务特别明显：如果模型多输出/少输出一个实体，会拖累后续的对齐，增加不必要的惩罚。

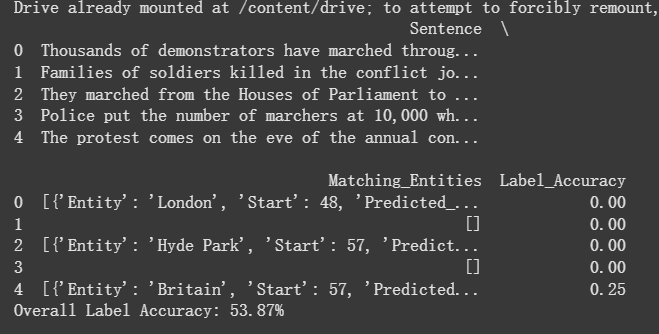
漏实体、多实体

我们希望模型完整、准确地输出目标实体列表。如果仅靠常规交叉熵，可能无法对“多预测”或“漏预测”这种实体级错误施加合适的惩罚。

需要混合策略

继续使用标准 Token-level 交叉熵（CE），因为它能引导模型学习到正确的语言序列基本形态和格式。

在输出完成后，将模型的最终生成解析成实体列表，与真实实体列表对比，对漏实体和多实体施加额外惩罚/负奖励。



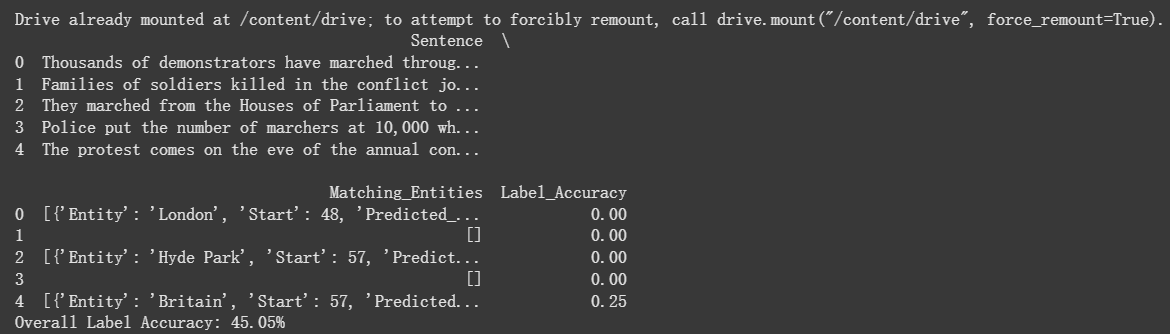
Spacy输出的位置不是token而是字符的位置（计入空格和标点），这和大部分数据集不一致，需要一些额外的处理，在目前的实验中，位置准确度并未得到计算。

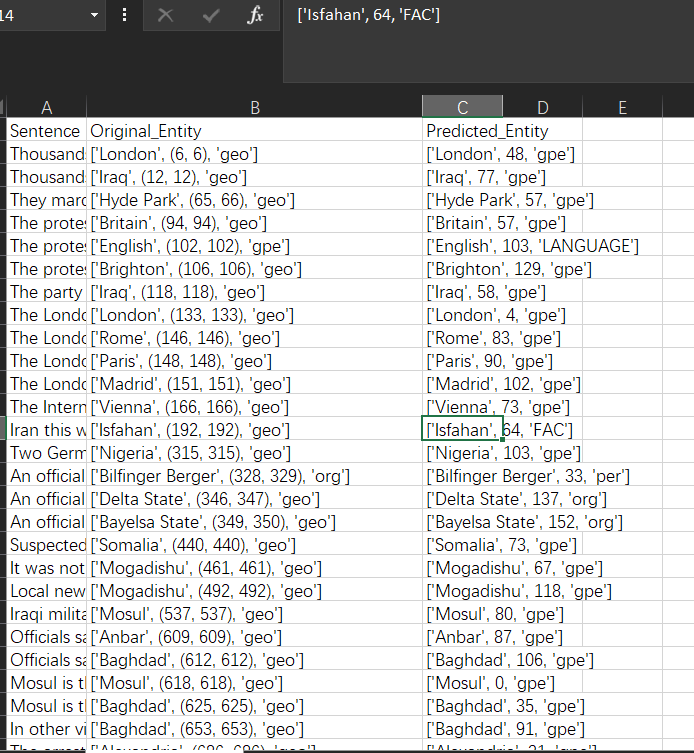
**1.2探究使用预训练管道结合特定数据集进行微调**

截取10000行数据进行训练测试集拆分(0.7:0.3)。在节省时间进行实验的情况下，训练均采用2 epoch.

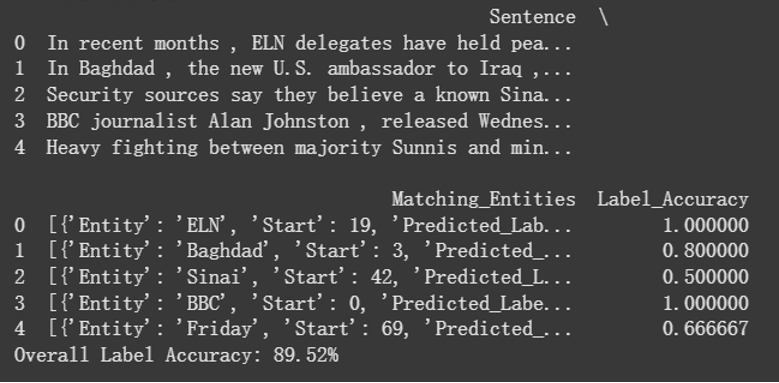
1.2.1使用空白管道进行训练

出现大量geo gpe的识别错误，如图

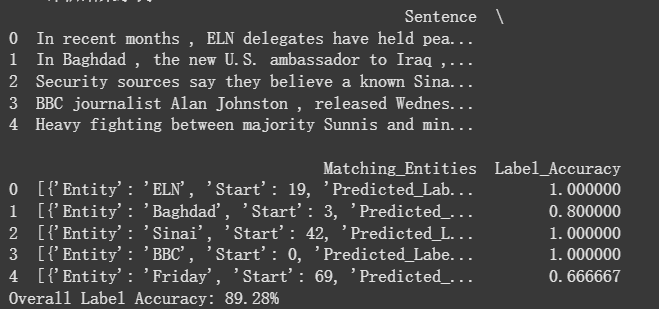


1.2.2载入预训练管道进行训练

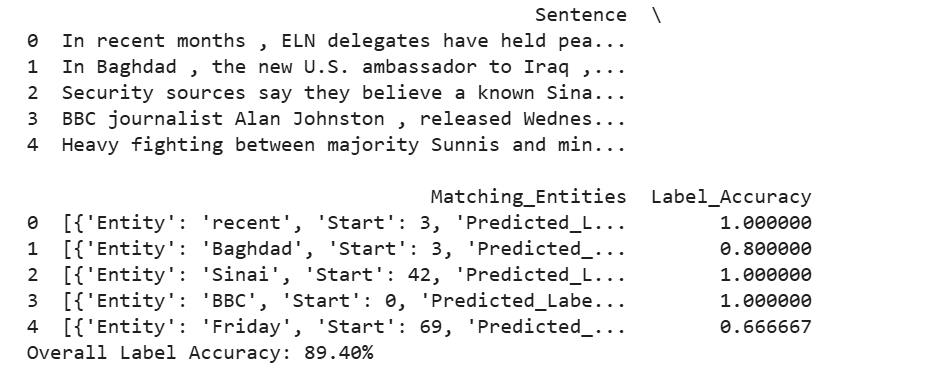
en\_core\_web\_sm



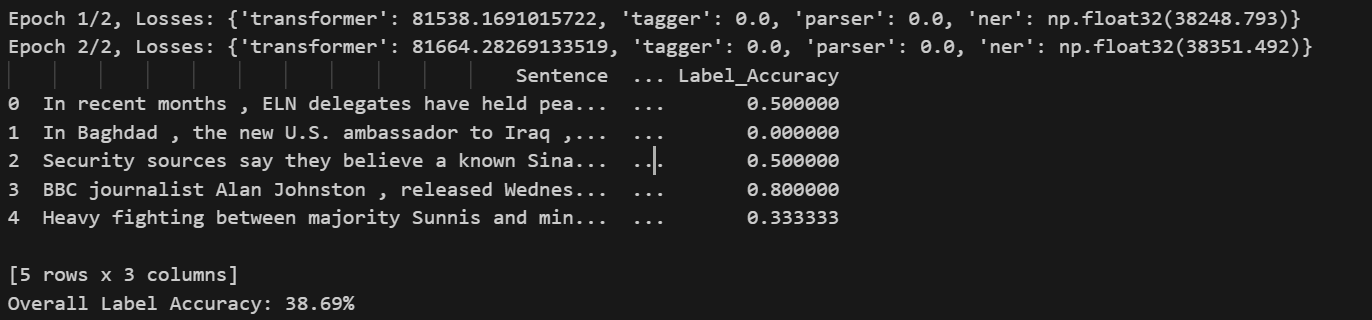
en\_core\_web\_md

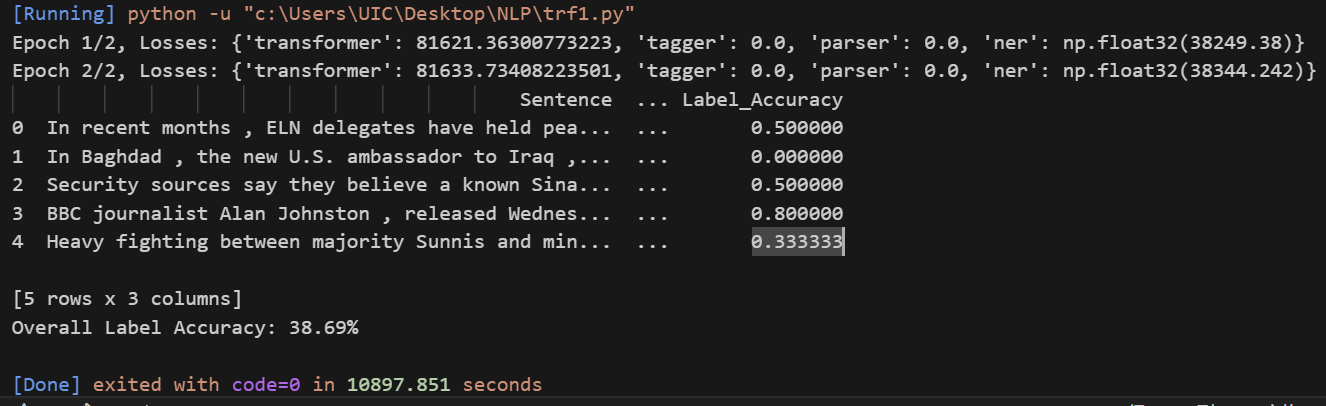


en\_core\_web\_lg

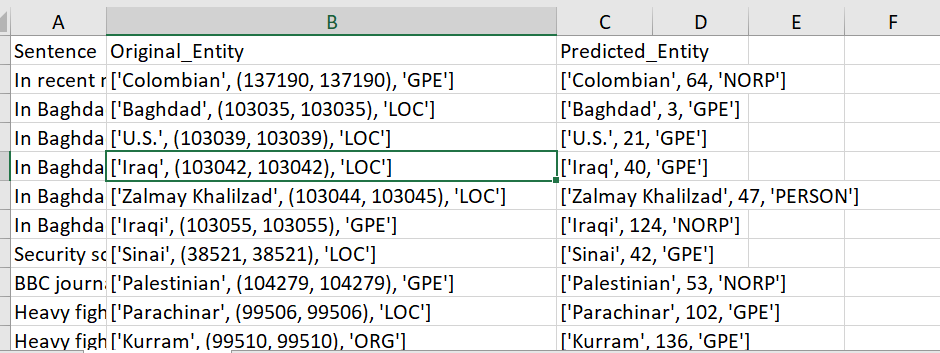


en\_core\_web\_trf





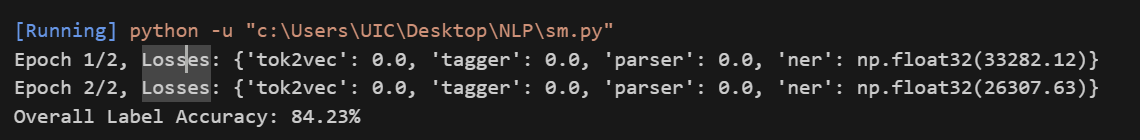
以上两个结果是在重构的trf测试代码和之前其他管道代码仅改写spacy.load()函数中管道的结果。这意味着代码有错误的可能很低，很有可能是trf管道不符合本次的实验条件，根据error\_df

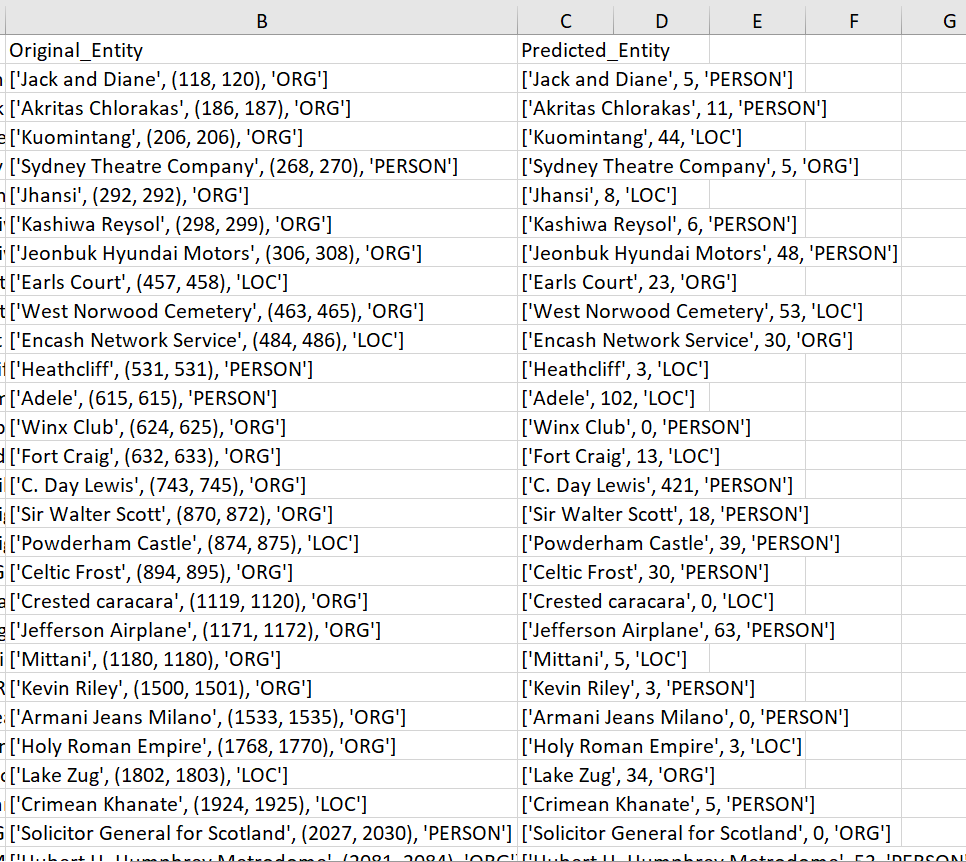
排查错误原因可能为trf的参数太多，训练epoch少和训练量少导致没有完全更新，trf的输出没有遵循要求的规则。

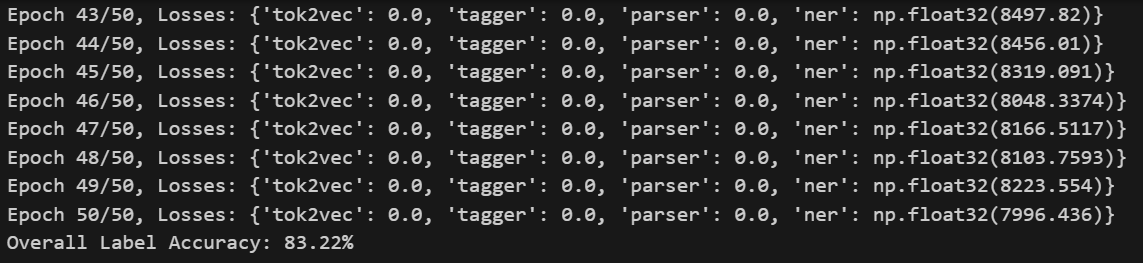
1.2.3 对微调管道进行更深入的研究

使用数据集1进行训练与测试，训练集和测试集使用原始数据集的train.conllu和test.conllu文件格式化而成。

在数据集1(数据标签为PER,LOC和ORG)中spacy的sm管道正确率，（2epouch）





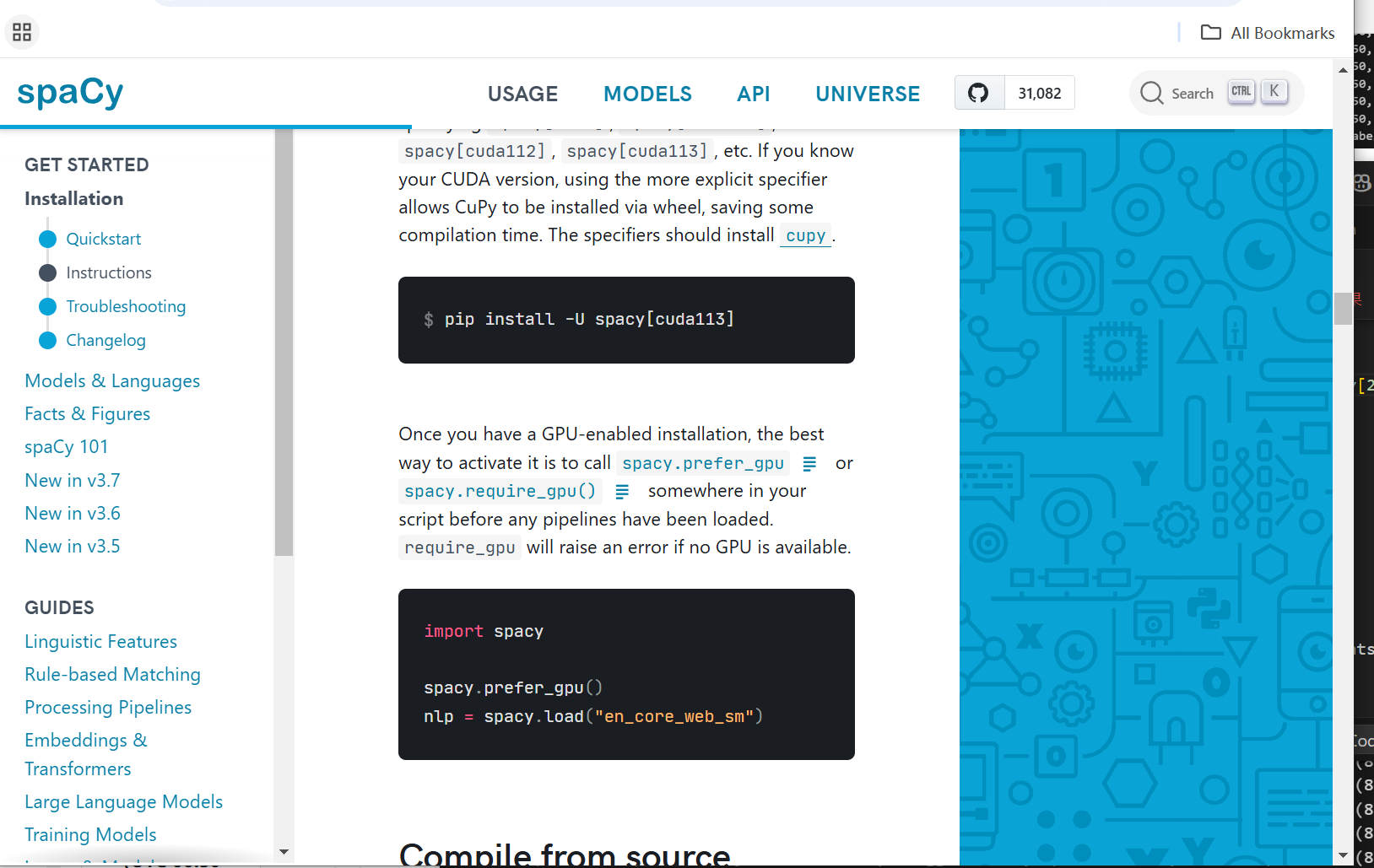
50epoch：

## **2.待解决问题：**

2.1 Deepseek

Deepseek api返回速度过慢，除了极长的思考时间，tokens的生成速度也过慢，约20tokens/s及以下。数据集每行tokens不统一，可能会造成溢出；目前采用比较保守的输入策略，造成测试较慢，同时api的返回不完整，会出现断行的情况。目前采用对api返回的sentence和数据集的sentence的前十个字符进行匹配的策略，依旧处理非常缓慢。

2.2 spacy

Trf管道的训练似乎是进行了封装，不能调用torch的相关cuda加速函数。在训练中出现了设备不统一的问题，主要出现在 spaCy 的 transformer 管道内部，由于该部分代码高度封装且缺乏公开的接口，很难通过外部修改来保证所有数据在训练时都统一在 gpu上。尽管spacy的官方文档中有相关使用gpu进行训练加速的说明

但未能解决问题。使用cpu训练这个管道过于缓慢。