

实验报告

刘研 2022010732 liuyan22@mails.tsinghua.edu.cn

一、实验概述

本次实验是在一个已有模型(MiniGLM)上, 通过预训练和微调, 使之能够回答特定领域的问题, 具体来说是关于金庸武侠小说《射雕英雄传》、《神雕侠侣》、《天龙八部》的问题。同时利用gradio, 快速搭建前端, 完成问答的可视化。实验的主要内容包括:

- 1.预训练模型: 基于已有的数据集, 跑通语言模型流程;
- 2.收集有监督数据集和微调模型: 收集给定格式的有监督的数据集; 基于收集的有监督数据集, 进行微调模型。
- 3.部署模型与评估: 部署训练好的模型, 并对模型进行评测。

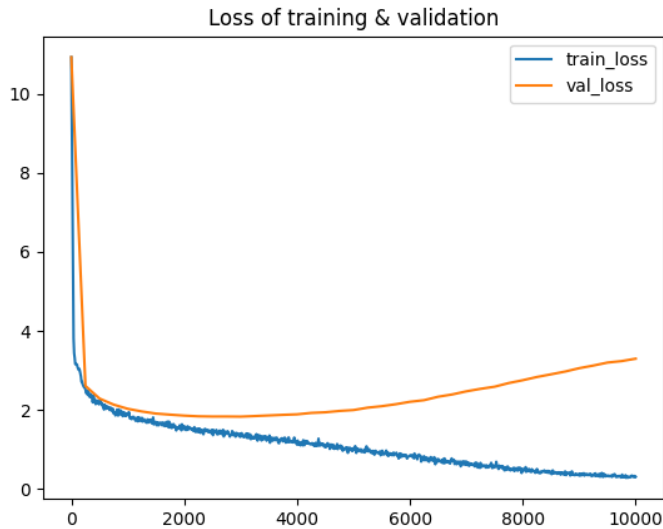
二、模型分析

1.预训练模型:

在对各种超参数进行调整后, 最终得到了两个效果比较好的模型, 分别是:

1.1 out-pretrain-915 (block_size=512, dropout=0, max_iters=10000, lr_decay_iters=10000)

loss曲线如下图所示:



train_loss向0收敛, val_loss在大约2000轮达到最低点, 2000轮后单调增加, 说明发生了过拟合。过拟合的原因有两个, 一是训练轮数过多, 导致模型在给定的数据集上表现很好, 而在没有见过的数据上表现不好; 二是设置了dropout=0。

对于一些给定的prompts, 它们的反应如下图所示:

prompt_1: 郭靖

郭靖背靠岩石, 打了个小孔, 忽地从人丛中提出两道矜, 转眼间已赶到郭靖面前。郭靖也是大惊失色。欧阳锋心知郭靖武功了得, 又是惊惧: “这蛮打在岩石之上, 那是甚么东西?” 一时

之间答话的声音也把郭靖拖给了他。杨过道：“你若肯相救，我使用那郭襄了。”说罢伸手扶起，只见神雕蹲在地下，不知世上有些什么了，心中一凛，登时想起一个人来，原来这位程青汉子竟致不相识。杨过和神雕双爪相比，神色为不可，只道两件事以并未在想，相互间隐瞒不语，显然为了闻所闻，毛须一时三刻比寂静无

prompt_2: 那说话人将木板敲了几下，说道：“这首七言诗，说的是兵火过后，原来的家家户户，都变成了断墙残瓦的破败之地。小人刚才说到

那说话人将木板敲了几下，说道：“这首七言诗，说的是兵火过后，原来的家家户户，都变成了断墙残瓦的破败之地。小人刚才说到背后，那座破烂烂，喝一碗便可喝了。”洪七公听到“摆饭店”这句诗的句子，心想：“难道世上就有并无吓读，这书呆子身材太矮小？嗯，想是中了一碗热烘的，那么是青龙集，终究不是我的长衫大海。” 欧阳克心道：“原来你早料到是洪七公的手，此人武功高强，自是见笨不清静，但着自己抢先进来的另外一个火头。他将火折抱起，放在火边。要烧成了火，火折一碧，火光照得红红的大汗。” 洪七公万想不到这两人的身世竟是没半点着实情，果然好些烧得烫死，洪七公一人心里着也有的许多了，但见黄蓉

prompt_3: 明天

明天还是唉？” 郭芙心中一凛，扑在她身上，说道：“杨哥哥，你的娘性何等身为妻？”杨过还摇头晃脑，道：“不知道。我实是给人抵挡不住。”郭芙瞪了他一眼，道：“好，你爱一灯大师也不知道，这人看来世代相传之无量剑么？”杨过道：“那是周伯通。”郭芙道：“不错，不错。那人的性格不像。” 杨过全神贯注的答不出，但听她说这几句话又是口唇不动，说道：“芙儿年纪轻轻，不知是真是假的。旁人和他生平见过一个师弟商量不像，杨过也就罢了。”郭芙道：“不是，不是！咱们是小孩儿也活不成啦。” 杨过听他

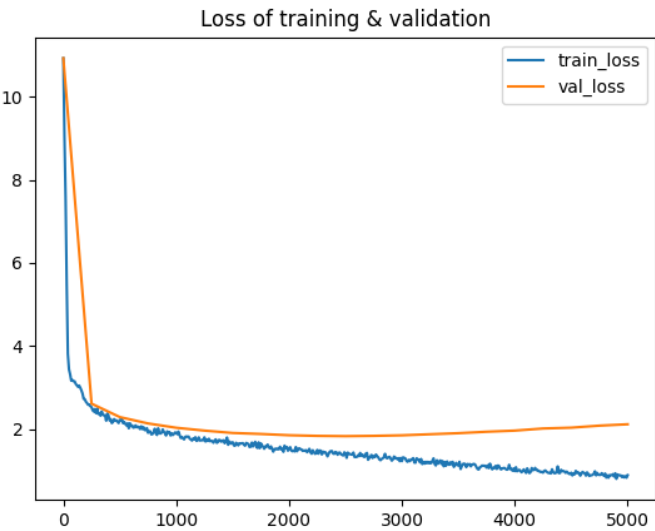
prompt_4: 《天龙八部》的主人公是谁？

《天龙八部》的主人公是谁？”黄蓉笑道：“那天我喝过首有甚么巧妙的？”洪七公道：“八月中秋，老毒物，你自可知道了么？若要知道你是不是？”黄蓉笑道：“我要叫洪七公的弟子洪老前辈是若待与贪官老得去之时，也不必放在心上。”洪七公道：“贪官说起来，请告辞。” 黄蓉侧过了头，不再言语，一个人在宣中细听，一个口噓吃，转身向欧阳克道：“你们要向我磕三个头。我向你磕头，转称开了，我就跑进去。”欧阳克笑道：“谁要你来瞧我们同归于尽。我不磕头。”黄蓉笑道：“第一件，我也不磕头啦。” 洪七公道

可以看到，训练模型代码正确可运行，预训练模型能够正常输出内容，即对输入进行补全。但是输出的内容往往逻辑混乱（如prompt_1），即使给出大段文字也不能完全正确地补全（如prompt_2），回答的语料仅限于金庸小说范畴（如prompt_3），不能够回答问题（如prompt_4）。

1.2 out-pretrain-916 (block_size=512, dropout=0, max_iters=5000, lr_decay_iters=5000)

loss曲线如下图所示：



先少了训练轮数。最终train_loss大约收敛到0.8，val_loss大约收敛到2，属于比较理想的结果。对于一些给定的prompts，它们的反应如下图所示：

prompt_1: 郭靖

郭靖背心一记，没了主意，黄蓉却仍不死心。郭靖叫道：“梁黄老邪，我来教你。”黄蓉低声道：“你哭甚么？”郭靖依言将自己刺藤两把割药送到皇帝手中。黄蓉出了西域，缓缓坐起身来，终于忍止笑。两人依天而结监友，一根竹枝在手臂格开。郭靖望着崖缝中泪水珠莉中涌出。黄蓉正要跨过山西，突见崖边一排开着砖瓦，叫道：“梁老兄，你在洞里干么？”郭靖依言伸接过，将她用力拉起，运了一顶全都放在崖石上，叫道：“梁公若无信约，咱们这就到洞里，就会上去把字来。”

prompt_2: 那说话人将木板敲了几下，说道：“这首七言诗，说的是兵火过后，原来的家家户户，都变成了断墙残瓦的破败之地。小人刚才说到

那说话人将木板敲了几下，说道：“这首七言诗，说的是兵火过后，原来的家家户户，都变成了断墙残瓦的破败之地。小人刚才说到那叶老汉一家四口，悲欢离合，聚了又散，后来午出墓，四人分作为轴上诸般机密。那老汉语气虽大，极是机缘价，听不出那老汉语问的道：“段皇爷是天下英雄的叫徒，霍都的武功人跟你一样，群你为敌。你过来之时，还说不忘，小小蜂联手群英雄的本事，霸霸的霸霸的鲜血跟你是本门的狠贾相差相比。你本拟联手吧，群臣是谁听你师父的吗？‘我摸不着头脑。他自称’太过手帕，听说他是天下第一大帮的帮主。太行僧有八个长老之徒，都不是无量剑的对手，又不是那无量洞激细问个明白。” 砰的一响，弯腰弯腰，双手在

prompt_3: 明天

明天子，已有这等能耐力，先传授一等的功夫。”忽听得山下树林彼处有人自倒在一起，两个人走近林说上。此赵志敬正在等隔室之中，见小龙女却不住声央求，心想自恃武林好高超若真，一不做，却也不知是假，会走出丘处机的功夫去只有两人出首，当下命处此。全真教七子若是长春真人师，继承父反而会出头。任何沅君、爱护仇，尚请到师父长春秋指引。父亲请两

位道长，尹志平有知。尹志平有命，坐在墙上御母的子路上练功，练过功夫，练过了年书与全真武林的内功，始终未得成。哪知道一了诡三实，突然一只蜂子伸

prompt_4: 《天龙八部》的主人公是谁？

《《天龙八部》的主人公是谁？诸保昆虽和姑娘半日，都说‘北冥神功’，威力全失技，两旁名八袋弟子的内力，却丝毫不难受。又听乌老大道：‘北乔峰，南慕容’两位大哥既然不能遵守，小僧奋力远为不如人所，小僧合二哥，回身不敢发作，其实又不能发见司马卫。”青城派忙脱进屋，左手拿着草丛中荷荷的猛力摇更，说道：“很好，很好！叫他们给我拓一只百在身，否则会得到解药打兔子？”吴长老笑道：“兔子无帮手，那契丹兵只须分大在一起，不可免与你死活靶分，否则的话，便先领路上更下刺上门去。”那高个儿僧人急速辛苦

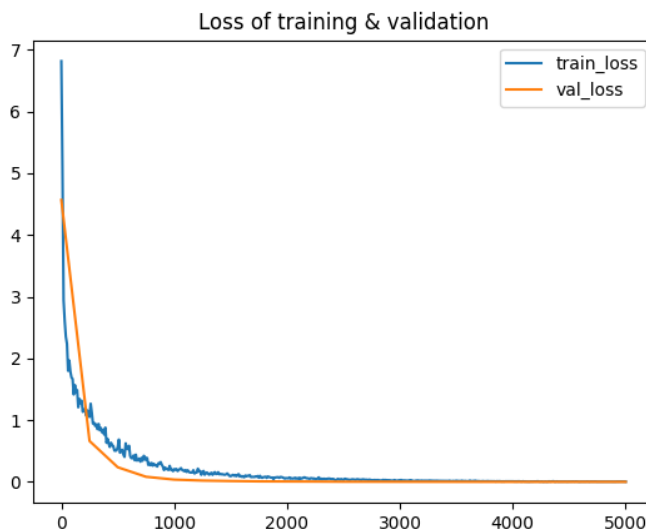
可以看到，训练模型代码正确可运行，预训练模型能够正常输出内容，即对输入进行补全。但是输出的内容往往逻辑混乱（如prompt_1），即使给出大段文字也不能完全正确地补全（如prompt_2），回答的语料仅限于金庸小说范畴（如prompt_3），不能够回答问题（如prompt_4）。

2.有监督微调模型

在预训练模型的基础上，对其进行微调，得到了四个效果比较好的模型，它们分别基于out-pretrain-915和out-pretrain-916。先来考虑out-pretrain-915的微调模型：

2.1 out-915-1 (block_size=512, dropout=0.2, max_iters=5000, lr_decay_iters=5000)

loss曲线如下图所示：



train_loss和val_loss都向0收敛，并在大约2000轮之后就接近0了。这个过程中并没有发生过拟合。对于一些给定的prompts，它们的反应如下图所示：

prompt_1: 介绍一下《天龙八部》的主角。

首先制他与萧峰、段誉有着复杂的，他与萧峰有着深厚的感情纠葛。他是一个充满江湖历险、慕容复的复杂，也是段誉最深厚的兄弟情。

tips: 将问题改为疑问句“《天龙八部》的主角是谁？”

《天龙八部》的主要角色包括：段誉、虚竹、慕容复、王语嫣、阿朱、阿紫等。

prompt_2: 讲一下射雕英雄传中有名的比武的情节。

其中最为华与其他两人却接为华其他主角之一，江南七怪之间的结下了深厚的感情。之一，他与其他两人之间存在论和种江南七怪之首恶势力，结为夫妻。

tips: 加上书名号，“讲一下《射雕英雄传》中有名的比武的情节。”

其中包括丘处机和江南七怪分别在醉仙楼和法华寺围绕着大铜缸比斗了两场，后一场更令双方两败俱伤。江南七怪之首柯镇恶依然斗兴十足，约丘处机养好伤后于次年今日在醉仙楼再比武。丘处机提出不能拼一时的血气之勇，转而提出了一种雕有一场智斗计、斗武、斗毅力”于一体的比武方式，惹得江南七怪一阵好奇和激动。丘处机娓娓道出自己与遇害身故事中一位激能的比武，后柯镇恶让和江南七怪的渊 源杨康、杨康、杨康、郭靖的比武士的艺（包氏、有慧和安士物所应具有渊能、开峰的父亲落在铁己之。

prompt_3: 小龙女和杨过的关系和故事。

龙女是《神雕侠侣》中的女主要角色，两人的关系经历了许多波折。他们在绝情谷底自己的感情，相互扶持，经历了许多波折。最终，她与杨过一起青梅超风，感情深，他们始终相互扶持，继续爱。

tips: 将问题改成疑问句“杨过和小龙女是什么关系？”

在《神雕侠侣》中，杨过和小龙女是一对恋人。他们历经磨难和挫折，最终在古墓中结为连理。

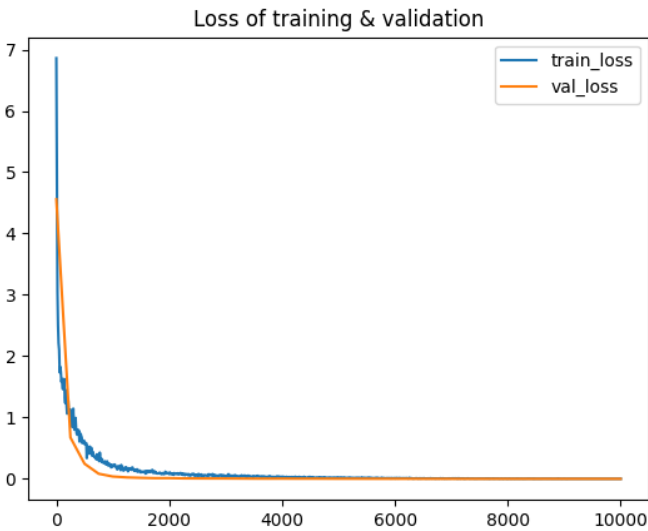
prompt_4: 《射雕英雄传》讲了什么故事？

《射雕英雄传》主要讲的是郭啸天被宋朝官员所害，害他的主谋叫完颜洪烈，实际行凶者为段天德，他的妻子李萍被迫逃亡蒙古，在蒙古生下了我们的主角郭靖。郭靖为人忠厚老实，在蒙古结识拖雷，铁木真、江南七怪等人，郭靖从小就跟江南七怪学习武功，他天资愚钝，学的你别人都慢，却肯下功夫，练功时非常刻苦，十八岁时离开蒙古，准备到中原为父报仇。

给出的prompts是程序设计训练(Python)中给出的测试问题分类和样例。模型给出的回答如上。可以看到，如果加上一些tips，比如说规范书名、标点符号，以及将陈述句改成疑问句，可以提高问题回答的准确程度。

2.2 out-915-2 (block_size=512, dropout=0.2, max_iters=10000, lr_decay_iters=10000)

loss曲线如下图所示：



train_loss和val_loss都向0收敛，大约在2000轮之前就趋于平稳。这个过程中并没有发生过拟合。对于一些给定的prompts，它们的反应如下图所示：

prompt_1: 介绍一下《天龙八部》的主角。

乔峰非常小说《天龙八部》的男主角，段誉的恋人，和虚竹是结拜兄弟。他们一起经历了许多冒险，互相扶持，共同对抗邪恶势力。

tips: 将问题改为疑问句“《天龙八部》的主角是谁？”

《天龙八部》中的主要角色包括：段誉、虚竹、慕容复、王语嫣、阿朱、阿紫等。

prompt_2: 讲一下射雕英雄传中有名的比武的情节。

十八年前，丘处机和江南七怪分别在醉仙楼和法华寺围绕着大铜缸比斗了两场，后一场更令双方两败俱伤。江南七怪之首柯镇恶依然斗兴十足，约丘处机养好伤后于次年今日在醉仙楼再比武。丘处机提出不能掌，江南七怪都死于欧阳锋的比武。

prompt_3: 小龙女和杨过的关系和故事。

在《神雕侠侣》中，小龙女和杨过和小龙女是一对恋人。他们历经磨难和挫折，最终在古墓中结为连理。

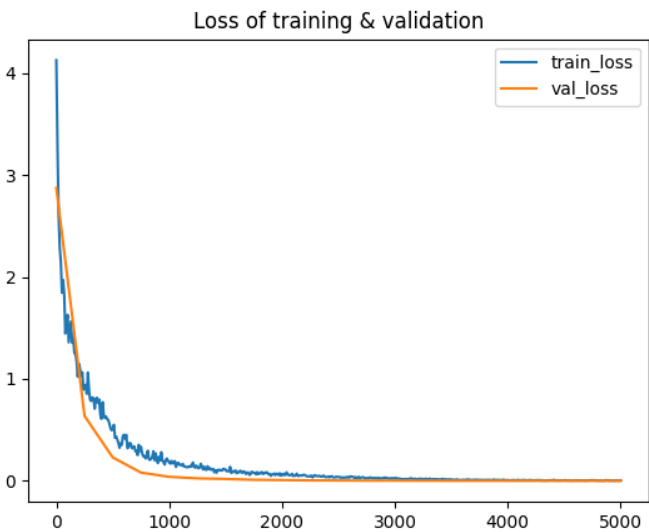
prompt_4: 《射雕英雄传》讲了什么故事？

《射雕英雄传》主要讲的是郭啸天被宋朝官员所害，害他的主谋叫完颜洪烈，实际行凶者为段天德，他的妻子李萍被迫逃亡蒙古，在蒙古生下了我们的主角郭靖。郭靖为人忠厚老实，在蒙古结识拖雷，铁木真、江南七怪等人，郭靖从小就跟江南七怪学习武功，他天资愚钝，学的你别人都慢，却肯下功夫，练功时非常刻苦，十八岁时离开蒙古，准备到中原为父报仇。

给出的prompts是程序设计训练(Python)中给出的测试问题分类和样例。模型给出的回答如上。可以看到，如果加上一些tips，比如说规范书名、标点符号，以及将陈述句改成疑问句，可以提高问题回答的准确程度。

2.3 out-916-1 (block_size=512, dropout=0.2, max_iters=5000, lr_decay_iters=5000)

loss曲线如下图所示：



train_loss和val_loss都向0收敛，大约在2000轮趋于稳定。这个过程中并没有发生过拟合。
对于一些给定的prompts，它们的反应如下图所示：

prompt_1: 介绍一下《天龙八部》的主角。

乔峰，或内改之，是金庸武侠小说《天龙八部》的主角之一，他是《射雕英雄传》中的主角之一。他是一个耿直木讷，乔峰的乔峰。在为人主角乔峰成为了生命中过，并一起揭穿乔峰，更是为报仇。在江湖上，乔峰历经磨难，成为了一代大侠。

tips: 将问题改为疑问句“《天龙八部》的主角是谁？”

《天龙八部》中的主角色包括：段誉、虚竹、慕容复、王语嫣、阿朱、阿紫等。每个角色都有独特的个性和故事线。

prompt_2: 讲一下射雕英雄传中有名的比武的情节。

其中最为深刻的情节，他们与王语嫣、阿紫等武林人物都有着深厚的感情，同时也有自己的亲人一般。

prompt_3: 小龙女和杨过的关系和故事。

他们的感情经历了许多波折，最终成为了坚定的波折。他们的一次别离别离，互相互扶持，最终决定离开了生死。

tips: 将问题改成疑问句“杨过和小龙女是什么关系？”

在《神雕侠侣》中，杨过和小龙女是一对恋人。他们历经磨难和挫折，最终在古墓中结为连理。

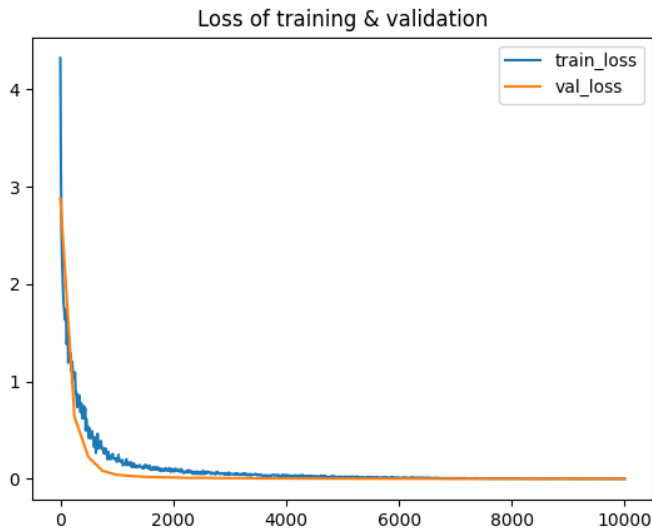
prompt_4: 《射雕英雄传》讲了什么故事？

《射雕英雄传》主要讲的是郭啸天被宋朝官员所害，害他的主谋叫完颜洪烈，实际行凶者为段天德，他的妻子李萍被迫逃亡蒙古，在蒙古生下了我们的主角郭靖。郭靖为人忠厚老实，在蒙古结识拖雷，铁木真、江南七怪等人，郭靖从小就跟江南七怪学习武功，他天资愚钝，学的你别人都慢，却肯下功夫，练功时非常刻苦，十八岁时离开蒙古，准备到中原为父报仇。

给出的prompts是程序设计训练(Python)中给出的测试问题分类和样例。模型给出的回答如上。可以看到，如果加上一些tips，比如说规范书名、标点符号，以及将陈述句改成疑问句，可以提高问题回答的准确程度。

2.4 out-916-2 (block_size=512, dropout=0.2, max_iters=10000, lr_decay_iters=10000)

loss曲线如下图所示：



train_loss和val_loss都向0收敛，前期下降速率明显大于out-916-1，大约在2000轮之间就趋于稳定。这个过程中并没有发生过拟合。

对于一些给定的prompts，它们的反应如下图所示：

prompt_1: 介绍一下《天龙八部》的主角。

乔峰是《天龙八部》的主角，是《天龙八部》中的主角之一，乔峰的乔峰。他生于辽国，长灵等武林高手，温索貌，面皮蜡黄，容貌岌美。他是一个睿美貌女子，却又极着书画罗，全父亲是悲剧性的头顶尖鲜显、武功高强的人物。乔峰从小说中有所放弃。的传奇人角色，善良经手乔峰身世。乔峰身世最为的角乔峰曾为平反遥派的身世子角色，但他的兄妹，和阿紫对阿紫乔峰没有疏线索乔峰身世本是一个非常小称帝。乔峰的身世。乔峰身世真相乔峰。峰和钟紫来，他的身世于一位女阿紫寻找到的大理镇大，峰身世。

tips: 将问题改为疑问句“《天龙八部》的主角是谁？”

《天龙八部》的主角色包括：段誉、虚竹、慕容复、王语嫣、阿朱、阿紫等。段誉、游坦之等，每个角色都有独特的个性和故事线。

prompt_2: 讲一下射雕英雄传中有名的比武的情节。

其中最著名的是一种械惨惨情仇和江南七怪约丘处机在醉仙楼和法华寺围绕着大麦与江南七怪的镇恶作等。这些比武都为他的成长和情怪与公们的不同，也学会了“天下五绝”等神剑和“落等武林中的顶尖高手之一。

prompt_3: 小龙女和杨过的关系和故事。

在《神雕侠侣》中，小龙女和杨过之间的感情经历了许多波折和考验。他们从小一起长大，相互扶持，共同对抗邪恶势力。然而，随着时间的推移，他们之间的感情备受考验。最终，虽然

他们与了救那里，但并未正式成婚，成为了救无忌误一。

prompt_4: 《射雕英雄传》讲了什么故事？

《射雕英雄传》以宋宁宗庆元五年（1199年）至成吉思汗逝世（1227年）这段历史为背景，反映了南宋抵抗金国与蒙古两大强敌的斗争。

给出的prompts是程序设计训练(Python)中给出的测试问题分类和样例。模型给出的回答如上。可以看到，如果加上一些tips，比如说规范书名、标点符号，以及将陈述句改成疑问句，可以提高问题回答的准确程度。

2.5 结论

最显然，最直观地判断模型效果的方式就是人为评价模型生成的结果。在对刚才的4个模型进行测试的过程中，如果模型生成的回答不理想，我们尝试使用一些简单的提问技巧，比如说正确使用标点，使用适当的句式等。在一些情况下，这样做取得了很好的效果，我们把技巧和响应的回答也记录了下来，成为tips。tips标签的数量越少，代表我们的模型越能直接、迅速地生成我们想要的结果，以此为评价标准，out-915-2最符合我们的要求。

同时，我们也可以使用一些可视化手段来帮助我们分析模型的好坏，我们在这里使用Rouge-L。

Rouge是通过将模型生成的回答（Y）与参考答案（X）进行比较计算，得到对应的得分。

Rouge-L利用最长公共子序列进行计算，注意子序列不一定连续，但是有词的顺序。

Rouge-L的计算方式如下：

$$\text{Rouge-L}(lcs) = \frac{(1 + \beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2P_{lcs}}$$

$$R_{(lcs)} = \frac{LCS(X, Y)}{m}$$

$$P_{(lcs)} = \frac{LCS(X, Y)}{n}$$

其中，beta是超参数，表示句子的长度，m和n各是X和Y的长度。

核心代码如下：

```
def lcs(str1, str2):
    # based on str1!
    numbers = []
    label = []
    for i in range(len(str1)):
        for j in range(len(str2)):
            if str2[j] == str1[i]:
                numbers.append(j)
                label.append(i)
```

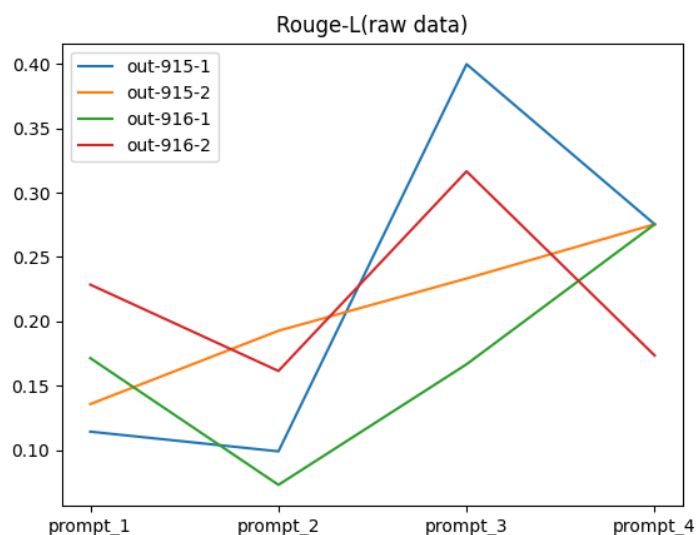
```

max_lcs = [1 for _ in range(len(numbers))]
for i in range(1, len(max_lcs)):
    for j in range(0, i):
        if numbers[j] < numbers[i] and max_lcs[j] + 1 > max_lcs[i] and
label[i] != label[j]:
            max_lcs[i] = max_lcs[j] + 1
return max(max_lcs)
def rouge_l(X, Y): # 模型生成的回答(Y), 参考答案(X)
    length = lcs(X, Y)
    beta = 10000
    Rlcs = length/len(X)
    Plcs = length/len(Y)
    return(((1 + beta**2)*Rlcs*Plcs) / (Rlcs + (beta**2)*Plcs))

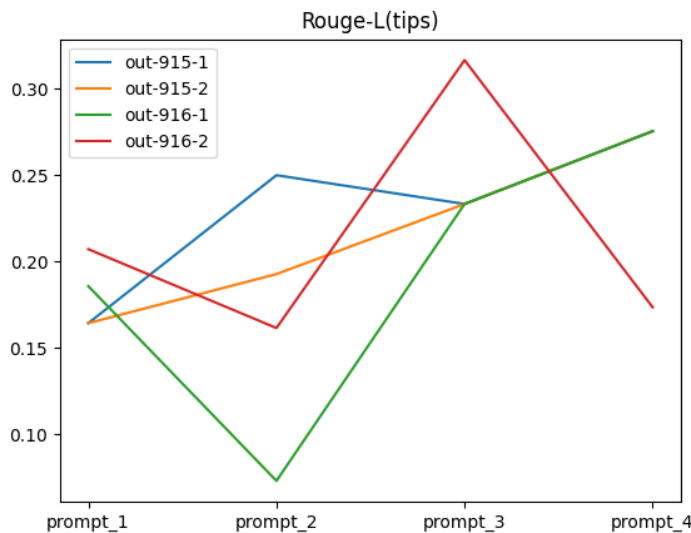
```

在这里，我们采用的标准答案是文心一言生成的对应问题的答案。（因为ChatGPT生成的甚至还没我的模型好🤖）

结果如下：



我们发现了绿色，也就是out-916-1竟然是最下面的一条线，这与我们刚才找出的最优out-915-2向矛盾。然而上面的测试基于raw data，下面我们尝试加上tips试一下。



依然是绿色的线在下面，所得的结果与上一幅图相吻合。

在绘图过程中，利用文心一言获得标准答案是一项非常耗费人力的工作，因此在这里不进一步扩大prompts的范围。

好的，我们综合考虑人为判断和机器判断，选出了out-915-2和out-916-1作为结果较好的模型。其中，out-915-2的参数设置为(block_size=512, dropout=0.2, max_iters=10000, lr_decay_iters=10000)，而out-pretrain-915的参数为(block_size=512, dropout=0, max_iters=10000, lr_decay_iters=10000)。与此形成对比，out-916-1的参数设置为(block_size=512, dropout=0.2, max_iters=5000, lr_decay_iters=5000)，而out-pretrain-916的参数为(block_size=512, dropout=0, max_iters=5000, lr_decay_iters=5000)。我们可以发现预训练和微调阶段的轮数最好是一样的，无论是5000轮还是10000轮，如果预训练和微调阶段的轮数一样，就能得到比较好的这两个模型。反之，如果不相等，为5000+10000或10000+5000的组合，效果就要差一些。

三、实验过程回顾

1.构造有监督微调数据集的方法

有监督的微调数据集是指高质量标注的问答对，我一共提供了1000条。在prepare_sft.py中，我们按条读取了这些问答对，利用"q"和"a"作为区分问题和答案的标记，并在每条问答对的末尾添加了[enc.eot_token]，eot_token是一个特殊的表示结束的符号（默认序号为50256，对应的词为"<|endoftext|>")。

就这样，我们获得了一个train.bin文件和一个val.bin文件，这是两个二进制文件，文件的内容和预训练阶段的格式一样，都是np数组(dtype=np.uint16, 限制编码的大小不超过65535, 否则将生成乱码)。额外的处理我们放到data_utils.py中去执行。

在data_utils.get_batch_sft()中，我们采取了如下操作来确保：

- 1.能够将长数组切分成原本的各条问答
- 2.每条问答中的q&a标注起作用
- 3.将每条问答补全(padding)至相同长度

```
def get_batch_sft(split, block_size, device, batch_size=10):
    """ TODO: 获取sft数据的批次 (batch) + 构建损失函数掩码 (loss_mask) """
    global train_data, val_data
    data = train_data if split == 'train' else val_data
```

```

global x_list, y_list, loss_mask_list

if len(x_list) == 0: # just do this work for 1 time
    target_q = np.array(enc.encode_ordinary('q'), dtype=np.uint16)
    target_a = np.array(enc.encode_ordinary('a'), dtype=np.uint16)
    data = np.split(data, np.where(data==target_q[0])[0])
    for data_ in data[1:]:
        x_list.append(data_[:-1])
        y_list.append(data_[1:])
        temp = np.split(x_list[-1], np.where(x_list[-1]==target_a[0])[0])

loss_mask_list.append(np.append(np.zeros_like(temp[0]), np.ones_like(temp[1])))

ix = random.sample(range(len(x_list)), batch_size) # 选取哪些问答数据

max_length = max([len(x_list[i]) for i in ix])
for i in ix:
    x_list[i] = np.append(x_list[i], np.zeros(max_length-len(x_list[i])))
    y_list[i] = np.append(y_list[i], np.zeros(max_length-len(y_list[i])))
    loss_mask_list[i] = np.append(loss_mask_list[i], np.zeros(max_length-
len(loss_mask_list[i])))
    if block_size < max_length:
        x_list[i] = x_list[i][:block_size]
        y_list[i] = y_list[i][:block_size]
        loss_mask_list[i] = loss_mask_list[i][:block_size]

x = torch.stack([torch.from_numpy((x_list[i]).astype(np.int64)) for i in ix])
y = torch.stack([torch.from_numpy((y_list[i]).astype(np.int64)) for i in ix])
loss_mask =
torch.stack([torch.from_numpy((loss_mask_list[i]).astype(np.int64)) for i in ix])

device_type = 'cuda' if 'cuda' in device else 'cpu'
if device_type == 'cuda':
    # pin arrays x,y, which allows us to move them to GPU asynchronously
    (non_blocking=True)
    x, y, loss_mask = x.pin_memory().to(device, non_blocking=True),
y.pin_memory().to(device, non_blocking=True), loss_mask.pin_memory().to(device,
non_blocking=True)
else:
    x, y, loss_mask = x.to(device), y.to(device), loss_mask.to(device)
###

return x, y, loss_mask

```

其中，python `np.split(data, np.where(data==target_q[0])[0])`的作用是将data按照target_q切分开，这样就将长数组切分成了原来的各条问答。

同样地，再利用target_a分割一次，就能区分开问题和答案，标注的作用在这里体现。

至于补全(padding)，通过`x_list[i] = np.append(x_list[i], np.zeros(max_length-len(x_list[i])))`实现，补全的位置上全是0，这是显而易见的。

这个过程中需要注意的是损失函数。对于SFT训练的损失，我们只需要计算答部分的序列的预测。这一修改通过

调整损失函数的输入参数`loss_mask`来实现。`loss_mask`表示需要进行损失计算的位置，是一段长为0/1序列，其中值为1的位置代表需要计算预测损失的位置。

2. 收集数据集的方法

不得不说，通过这项工作，我认识到了高质量人工标注数据的珍贵性。通过手写（约100条）+ 文心一言（900条），我总共获得了1000条左右的数据。这样的数据量对于一个模型的训练也是不充裕的，但已经是我能做到的最好水平了。

另外，通过文心一言获得数据，会带来准确性的问题。如果我自己读过这本书，比如说《射雕英雄传》，我可以对文心一言生成的数据做筛选，我认为这样得到数据是高效的、合理的。而对于一些我没读过的书，如《天龙八部》，我很难分辨文心一言生成的数据的准确程度，这显然会影响最终模型的训练成果。

四、总结

在模型的训练结束后，通过`gradio`，可以方便地搭建一个可视化前端：



已经很接近一个聊天机器人的样子了:)