task1 simple RAG实现

一、参考资料

代码上传到了：https://github.com/TimingWu/RAGchallenge

大模型（LLMs）simple_RAG 实现篇 https://articles.zsxq.com/id_roqf2ge4c1b3.html

任务说明：先把这个项目跑起来，把基本逻辑搞懂，然后后面再一步步优化

任务截止时间：20240502（本周四）下午8点

二、环境搭建

1. win下安装faiss-gpu，参考 https://github.com/facebookresearch/faiss/blob/main/INSTALL.md，我这里会安装1.7.4版本的faiss

```
conda install -c conda-forge faiss-gpu
```

2. 模型下载

```
git clone https://huggingface.co/Qwen/Qwen1.5-4B
git clone https://huggingface.co/BAAI/bge-large-zh-v1.5
```

解决hf 443 max retries或timeout：把端口配置为梯子使用的端口

```
git config --global http.proxy 127.0.0.1:xxxx
git config --global http.sslVerify false
```

三、数据构建

准备简单的从百度百科上弄些数据下来，因为只是txt文件的话，还是手动复制比较好，有些表格的就不复制了，然后去除了“播报” “编辑”这样的会占一整行的信息。下面是效果：

基本信息
《星际穿越》是2014年美英联合制作的科幻电影，由克里斯托弗·诺兰执导，马修·麦康纳、安妮·海瑟薇领衔主演。
该片在物理学家基普·索恩的黑洞理论之上进行改编，主要讲述了一组宇航员通过穿越虫洞来为人类寻找新家园的冒险故事 [1]。
该片于2014年11月5日在美国公映，11月7日在英国公映，11月12日在中国大陆公映。并于2020年8月2日在中国大陆重映 [37]。
2015年，该片获得了第87届奥斯卡金像奖的五项提名，并获得最佳视觉效果奖 [39]。
2024年，该片为纪念上映十周年，宣布于秋季在北美重映。
剧情简介

地球农作物因气候转变及枯萎病而经常失收，曾是美国国家航空航天局的工程师和航天飞机驾驶员的库珀（马修·麦康纳饰）被迫成为农民以协助解决粮食危机。库珀的10岁女儿墨菲（麦肯基·弗依饰）发现其房间书架上的书本无故掉到地上，认为这是幽灵现象。不久后，一场沙尘暴在墨菲房间中留下二进制坐标，二人驱车到达坐标位置后发现那是北美空防司令部（美国国家航空航天局的秘密基地）。
基地负责人布兰德教授（迈克尔·凯恩饰）向库珀透露土星附近出现了虫洞，认为外星智慧有意协助人类前往遥远星系移居，总署在约十年前已派遣了十二名科学家穿越该虫洞，各自降落在多个被认为有居住可能性的行星上，传送回来的资料显示其中一个以黑洞“卡冈图雅（Gargantua）”为中心的行星系统有三颗星球可能适合殖民。行星以降落的科学家名字命名：米勒（Miller）、埃德蒙斯（Edmunds）和曼恩（Mann）。库珀答允布兰德教授的要求，担任航天器永恒号（Endurance）的驾驶员前往执行拉撒路项目（Lazarus mission）：A计划为确认星球适居性后，通过布兰德教授的重力方程协助地球人类前往殖民；B计划为带着多个人类胚胎进行殖民，留在地球的人类则会灭绝。墨菲因担心库珀一去不回而深感愤怒，库珀在二人没有好好道别的情况下离开，他与布兰德教授的女儿艾米莉亚（安妮·海瑟薇饰）、物理学家罗米利（大卫·吉亚斯饰）、地质学家道尔（韦斯·本特利饰）和两个机器人塔斯（TARS）与凯斯（CASE）前往太空登上永恒号启程。
众人先到达米勒星；“卡冈图雅”庞大引力造成的引力时间膨胀使米勒星的一小时约为地球的七年。但众人漏算了前几年收到来自米勒的数据以此星球的时间而言其实只是几小时前，因此除罗米利与塔斯外一众成员乘坐飞艇浸游者号（Ranger）降落星球后，才发现地表只有一片汪洋且经常出现巨型海啸，其海啸的来袭使道尔丧生，并且延误了回程。众人返回永恒号后发现对罗米利而言已度过了超过23年的时间。
众人认为只能选择前往余下两个星球的其中一个，经过一番争辩后终决定前往曼恩星。同时墨菲已成长为与库珀离开地球时相同的同龄，她加入了美国国家航空航天局协助布兰德教授解开拯救地球人类所需要的重力方程，但教授在健康恶化弥留之际向墨菲承认A计划只是个谎言，不可能实现。另一方面，科学家曼恩（马特·达蒙饰）被众人从人工睡眠中唤醒后表示，方程因缺乏黑洞引力奇点的数据而无法完成，因此永恒号的真正目的并非拯救地球人类，库珀因此十分愤怒与懊悔，打算放弃项目、回到地球寻找家人。在稍后的营地寻找任务中，曼恩破坏了库珀的太空面罩并表示所有宜居数据均为他一手创造的假数据，目的是希望国家航空航天局能派人前来救他。曼恩留下待死的库珀后夺取浸游者号前往永恒号，同时罗米利试图从被曼恩拆解的机器人奇蕾（KIPP）中取得探测资料，触动了陷阱引发爆炸身亡。艾米莉亚救回即将因吸入过多氨气而窒息的库珀，二人乘坐降落艇追赶曼恩。曼恩在与永恒号并未完全对接成功的情况下打开气闸，产生的失控减压使他身亡，并导致永恒号失控旋转，库珀成功对接永恒号使其稳定下来。
永恒号上的资源已不足以返回地球，因此二人驶向“卡冈图雅”，射出塔斯让它收集引力奇点的数据，在接收数据传送回地球后以重力助推前往埃德蒙斯星实行B计划。为减少永恒号的质量让艾米莉亚逃生，库珀让自己驾驶的浸游者号于耗尽燃料后分离并进入黑洞，但在浸游者号被毁弹射逃生后，库珀发现自己身处一个非线性流动的第五维度超正方体；库珀至此明白了未来超越了时空并进化到较高文明的人类创造了四维超正方体和虫洞以拯救过去的人类，也明白自己亦是墨菲儿时遇上的幽灵、引导自己参与拉撒路项目（这是一个命定悖论）。库珀把塔斯取得的黑洞数据以引力波透过墨菲的手表传送给她；成人墨菲在回忆此事时终于了发现父亲的讯息，她完成了布兰德教授的方程使人类得以离开地球，前往环绕着土星运行的空间站居住。
四维超正方体空间在数据传送完毕后关闭，库珀被送离虫洞回到土星附近，被土星宇宙殖民地人员救起。此时他已离开地球91年，理论上而言已是124岁。库珀在空间站上与年老垂死的墨菲重逢，但墨菲不欲父亲目睹自己离世而让他离开，并说服他前往埃德蒙斯星去寻找艾米莉亚。库珀与塔斯搭乘一艘次世代漫游者号前往埃德蒙斯星。 [1]
制作人　　Lynda Obst、克里斯托弗·诺兰、艾玛·托马斯
导演 克里斯托弗·诺兰
副导演（助理） 达里乌斯·德·安德拉德、盖瑞·哈维斯、布兰登·兰伯丁、麦肯芝·劳伦斯、小格里高利·J·珀利克、梅甘·M·香柯、Dillon Neaman、Valgeir Gunnlaugsson、Nilo Otero、Jason Z. Kemp
编剧 克里斯托弗·诺兰、乔纳森·诺兰
摄影 霍伊特·范·霍特玛
配乐 汉斯·季默
剪辑 李·史密斯
选角导演　约翰·帕帕席德拉

霸王龙属于暴龙超科的暴龙属，为该属下的唯一种，于1905年由美国古生物学家，美国艺术与科学院院士亨利·奥斯本描述命名。如果参考其它恐龙种名的翻译格式，它的种名翻译为“君王暴龙”会更合适。成年霸王龙体长约12米，体重7吨左右，是地球上有史以来最大的陆地捕食者之一。
霸王龙是久负盛名的恐龙，拥有海量的标本收藏，其中几十件标本都达到了非常高的完整度，使得古生物学家对于霸王龙的研究不仅仅局限于经典的形态描述，还可以对它的个体发育，生物力学，种群生物学等做出更详细的分析。如对于霸王龙头骨生物力学的研究表明它具有粉碎骨骼的咬合力，对它的脑颅三维重建表明霸王龙具有极佳的视觉和嗅觉，而对霸王龙个体发育序列的研究表明，霸王龙具有非常快的生长速度。很多在其它恐龙物种或是其它古生物身上无法落实的研究手段都可以用霸王龙尝试。在科学研究方面，霸王龙在国际顶级学术期刊《Science》上拥有属于自己的综述论文 [1]，并且与它相关的各种研究也会频繁登上《Nature》和《Science》这样的顶级学术期刊。虽然目前霸王龙的体型受到了大型异特龙类以及棘龙的挑战，但是综合研究深度以及文化影响力来看，它依然是名副其实的恐龙之王。
命名由来

“tyrannos”意为暴君，种名中的“Rex”为国王的意思，以凸显霸王龙巨大的体型。合并起来的意思就是残暴的蜥蜴之王
基本信息和形态学

霸王龙的外鼻孔是一个前后向延长的开孔，其前背侧缘由前颌骨构成，后缘背侧缘由鼻骨构成。眶前窝非常大，眶前窗和上颌孔都位于其内。眶前孔的前缘，腹缘和背缘由上颌骨构成，后背侧面由泪骨构成，后腹侧缘由颧骨构成。前颌孔是一个圆形的小孔，位于眶前窝的前缘。此孔在头骨的外侧面无法被看到。上颌孔是一个圆形的孔，完全处于上颌骨之内。眼眶呈背腹向增高，前后向压扁。其腹缘的宽度大于背缘的宽度。其前边缘由泪骨构成，后缘和背缘由眶后骨构成，腹缘由颧骨构成。下颞孔呈钥匙孔状。其背缘主要由鳞骨的前突构成，后缘由鳞骨和方颧骨构成，前缘和腹缘由颧骨构成。上颞孔的背视呈长方形。
前颌骨具有四枚牙齿，每一枚都呈圆锥状。前颌骨的牙齿横截面略微成“D”字形，牙的后表面比较扁平。一排小的滋养孔延平行于齿列的方向排列。前颌骨的升突为圆柱状，构成了外鼻孔的前缘。前颌骨的腭面呈拱形。两侧前颌骨的裂隙不算明显，但是犁骨的前突可以伸入到两块前颌骨之间。
鼻骨限于吻端的背面。两侧的鼻骨愈合，但是在吻端和后端略有分开。鼻骨在向后延伸的过程中向外侧扩展。鼻骨前端有一个小的向腹侧延伸的突起，与前颌骨相连接，将上颌骨从外鼻孔的边界排除出去。每块鼻骨具有两个后突，腹外侧突比较短，与泪骨相关节，背内侧突较长，与颌骨和前额骨相接。由于上颌骨与泪骨的关节，鼻骨没有参与眶前窗的形成。成年霸王龙的鼻骨背面具有粗糙的骨质纹饰。上颌骨向外侧扩展，齿列向腹缘突出。
上颌骨前侧的牙齿具有很深的牙根。上颌骨的牙齿在内外侧向上压扁，明显的向后弯曲，并且大于前颌骨的牙齿。上颌牙齿双侧都有锯齿。一排滋养孔在上颌骨的腹缘延平行于齿列的方向向后延伸。在上颌骨和前颌骨的交界处具有一鼻下孔。在第十二枚上颌骨齿以后，上颌骨开始向后变细，并最终与颧骨相接合。上颌骨的外侧面非常粗糙，但位于眶前窝的部分则非常光滑。一个漏斗形的骨质间隔将前颌孔和上颌孔相隔开。前颌窦占据了整个整个骨头的前五分之一，并通过一个三角形的窗与上颌窦相连通。上颌骨的腭骨突是一个薄的骨片，它构成了内鼻孔的前缘和外缘。
两侧的犁骨在内侧面联合，并在前部有一个扩张略呈菱形。犁骨的大部分都是细长的棒状，并将内鼻孔分开，它们膨大的位置正好位于前颌窦的位置。犁骨前部的膨大骨壁非常薄，但是相背面凹陷。在上颌窗位置后的部分，犁骨腹侧有一对钩突，犁骨与颚骨相关节的地方背腹向扩展。犁骨前侧扩展的位置具有容纳下颌齿的小凹陷。犁骨在颚骨升突后的部分开始变细，具有细的后突与翼骨的前缘相接。
颚骨的升突向上陡起并在前向和背向膨大，形成一个鸡蛋形状的腭骨突。颚骨具有气腔化构造。颚骨的后侧有一个背腹向纤细的翼骨突。颚骨的后缘的外侧形成了凹面用于与翼骨相关节。颚骨的腹面具有一大的结节，其位置在翼骨的前侧。
泪骨的降突将眶前窗和眼眶区分开。泪骨的前突构成了眶前窝的背边缘。强壮的后突构成了眼眶的上边缘。从背面看，泪骨呈半月形，并对鼻骨的后缘和颜骨进行了一定程度的挤压。泪骨的降突向前强烈的拱起。泪骨后突的背缘具有粗糙的骨质纹饰，与眶后骨的骨质纹饰相衔接，中间有一明显的凹口分开。泪骨窦向前开口。泪骨的前突呈二分支状，小的腹侧支与上颌骨连接，大的背侧支与上颌骨和鼻骨连接。
颧骨的腹缘几乎构成了颊部的腹缘，其后侧升突构成了眶后柱的一部分。在眼眶下的位置，颧骨的外侧面向外膨大。颧骨前侧升突构成了眶前柱的三分之一的高度。两个升突之间的凹陷非常深且窄。后侧升突的顶端非常尖锐，其后外侧面上有一个小的粗糙面可能用于与鳞骨关节。在后侧升突后面的部分，颧骨构成了下颞孔的腹缘。额骨部分愈合。
额骨前缘与鼻骨相关节的关节面非常复杂。额骨的背面整体来讲比较光滑和凹陷，额骨与眶后骨的关节面呈“V”字形。两侧额骨的腹面都有一个大的凹陷窝，用于容纳巨大的嗅叶。顶骨完全愈合，且两侧顶骨的中线处也没与切迹。顶骨额骨关节处的裂隙不甚清晰，但是在这一位置有一高高隆起的脊。顶骨构成了上颞孔的内侧表面。从后面看，顶骨的后面形成了一对翼状的凹面，每块顶骨具有一个尖细的腹外侧突将鳞骨与副枕突隔开。顶骨构成了中脑的顶面。
眶后骨的降突向前延伸进入到眼眶的里面。眶后骨构成了上颞孔的前边缘和外侧边缘。鳞骨与眶后骨的内侧重叠。眶后骨的背缘具有很粗大的骨质隆起。眶后骨的后侧突尖细。眶后骨的降支构成了眶后柱的上半截，且发育有一个前脊。一对小孔位于前脊之前的背侧面上。眶后骨降支与颧骨连接的地方隆入到眼眶内部，使得眼眶呈钥匙孔的形状。
从侧面看方颧骨呈沙漏型。方颧骨的后缘具有一个明显的凹陷，前缘具有两个突。背侧突与鳞骨相关节，且深入到下颞孔内部。腹侧突与颧骨相关节。前脊侧突的侧面具有明显的凹陷。前腹侧突的截面呈现出圆柱状。方颧骨的外表面具有两个非常明显的结节，第一个靠背侧，位于前脊侧突与前腹侧突之间的高度对应的外表面上。第二个位于前腹侧突的后缘。
在关节状态下，鳞骨的大部分是看不到的，只有背侧和前外侧突暴露的比较多。鳞骨的前外侧突是二分式的，在关节的状态下，只有上分支能够看的到，并一直延伸到眶后骨骨质粗隆的位置。鳞骨与顶骨的关联处位于上颞孔的后外侧角。从背面看，上颞孔在两块骨头关节处有一个明显的后侧凹陷。鳞骨的前突参与了上颞孔的外侧壁的构成。鳞骨的腹外侧边缘构成了下颞孔的背边缘。眶后骨的后突插入到鳞骨前突的一个很深的与之对应的插口里。鳞骨的降支与方颧骨向关联。一个很深的凹陷位于降支的后缘。鳞骨内侧面的后背侧角可以看到一个凹窝，用于与方骨关节。
方骨围绕成了后方骨孔的边界。方骨与下颌的关节头被一个沟分成两个部分，外侧髁大于内侧髁。方骨具有很发达的气腔化构造。

引用之类的标记就当噪声了，应该不会有影响。当然上面这两个可能在模型的训练数据中已经有了，所以又加入了一条最近的新闻：

华为发布最新一季度业绩之时，其管理层也发生人事变动。

4月30日，据36氪报道，华为内部当日发布人事调整文件，宣布余承东将卸任华为终端BG CEO一职，但仍保留终端BG董事长职位。原华为终端BG首席运营官何刚接任华为终端BG CEO。

界面新闻从多位华为人士处确认，此次调整属实。

余承东自2011年开始担任华为终端公司CEO，历时近13年。接棒者何刚则是早在1998年便加入华为，2012年出任华为消费者BG手机产品总裁。

他在任期间，华为终端业务收入从2012年的1601亿元，增长至2020年的巅峰4829亿元，旗下手机业务也曾一度登顶全球智能手机市场单季度销量首位。

一位华为人士对界面新闻分析，此举并不意味着余承东权力范围缩窄。

根据华为官网信息，目前余承东的职位分别是：华为常务董事、终端BG董事长、智能汽车解决方案BU董事长、智能终端与智能汽车部件IRB主任。

上述人士表示，担任终端BG董事长意味着余承东仍然是该业务的一把手。此外，华为IRB（投资评审委员会）部门的存在极为重要，其作用在于决定具体业务项目的投资立项。余承东仍然担任智能终端IRB主任，表明其对该业务具有投资决策权。

接近华为的人士对界面新闻分析称，此次余承东卸任，可能是要把更多重心放在汽车业务的信号。

事实上，余承东的工作重心已在逐渐转移。从去年下半年开始，他已不再管理一线手机业务，而是将更多工作交给何刚。

多位华为人士认为，何刚是理论上最适合接任余承东职位的人选。他曾参与并主导华为Mate及P系列的手机研发，这也是奠定华为在高端手机市场地位的两大系列。

何刚加入华为时最初负责无线业务。2011年，在余承东的推动下，何刚转战手机业务，出任华为消费者BG手机产品线总裁，此后升任华为终端BG首席运营官、华为终端BG可持续发展委员会主任，一直担任终端业务的二号人物。

华为终端业务此前曾遭遇了诸多困境，但从当日发布的财报来看，其已在制裁常态下找到了适合自己的求生之路。

财报显示，一季度华为实现营业收入约1784.5亿元，同比增长36.66%；归母净利润约196.5亿元，同比增长约564%，净利润率达11%。而净利润率的大幅提升，与华为终端业务的营收表现有直接关系。

在过去一年中，华为在手机领域动作频频，先是在8月，其高端旗舰Mate 60系列宣布回归，一时间引发了抢购热潮。这款手机备受关注的手机搭载了麒麟芯片，至今部分型号仍供不应求。到了12月，华为又发布nova 12系列。

华为最新推出的手机产品则是Pura70系列。而在新能源汽车领域，该公司也相继发布了智界新S7、问界新M5、享界S9等诸多新品。

此次人事调整后，无论是在手机终端还是在汽车领域，华为或将回归以往的激进打法，收复失地与开拓新市场同步进行。

诶嘿，已经等不及要测试下模型待会知不知道余总的新动态了！

四、代码及测试

1. 分句

1. 模型下载

采用split("\n")和spacy两种方式，spacy模型为 `zh_core_web_sm` ，使用前需先下载

```
python -m spacy download zh_core_web_sm
```

查了一下，这个模型本质上还是基于规则的，没有按语义进行分割的功能，但是可以做到把引号内的话都分到一句，按。等符号进行分句，已经省去不少自己写会遇到的麻烦了。另外spacy和nltk等工具比会快一点，这里也是用的较小的模型，效率高精度也会低一些，先用着了。

2. 代码

```
def process_file(file_path):
    with open(file_path, encoding="utf-8") as f:
        text = f.read()
        sentences = text.split('\n')
        return text, sentences

def process_file_with_spacy(file_path):
    nlp = spacy.load("zh_core_web_sm")
    with open(file_path, encoding="utf-8") as f:
        text = f.read()
        doc = nlp(text)
        sentences = [sent.text for sent in doc.sents]
        return text, sentences

def test_split(file_path):
    _, s1 = process_file(file_path)
    _, s2 = process_file_with_spacy(file_path)
    print(s1[:10], "\n"+"*"*50+"\n", s2[:10])
    print("*"*100)
    print(s1[-10:], "\n"+"*"*50+"\n", s2[-10:])
```

实际使用时的代码，另外补充下接收输入和读取文件的逻辑

```
def process_file(file_path, mode="spacy"):

    with open(file_path, encoding="utf-8") as f:
```

```
        text = f.read()
        match mode:
            case "spacy":
                nlp = spacy.load("zh_core_web_sm")
                doc = nlp(text)
                sentences = [sent.text for sent in doc.sents]
            case "split":
                sentences = text.split('\n')
        return text, sentences

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--doc_dir",
                        help="directory of documents",
                        default="/rag_documents")
    parser.add_argument("--emb_model",
                        help="the path to the embedding model",
                        default="F:/models/bge-large-zh-v1.5")
    parser.add_argument("--gen_model",
                        help="the path to the generative model",
                        default="F:/models/Qwen1.5-4B")
    parser.add_argument("--doc_number",
                        help="the number of relevant documents to use for
context",
                        default=1)
    parser.add_argument("--split_mode",
                        help="the mode to split documents into
sentences,including 'spacy' and 'split'",
                        default="spacy")
    args = parser.parse_args()

    print("Splitting documents into sentences...")
    documents = {}
    for idx, file in enumerate(tqdm(os.listdir(args.doc_dir))):
        cur_filepath = os.path.join(args.doc_dir, file)
        text, sentences = process_file(cur_filepath, args.split_mode)
        documents[idx] = {"file_path": file,
                          "sentences": sentences,
                          "document_text": text}

```

3. 效果测试

*号前都是split("\n")分的，后面是spacy，每组两张图，第一张前十句，第二章后十句。

星际穿越组

['基本信息', '《星际穿越》是2014年美英联合制作的科幻电影，由克里斯托弗·诺兰执导，马修·麦康纳、安妮·海瑟薇领衔主演。', '该片在物理学家基普·索恩的黑洞理论之上进行改编，主要讲述了一组宇航员通过穿越虫洞来为人类寻找新家园的冒险故事 [1]。', '该片于2014年11月5日在美国公映，11月7日在英国公映，11月12日在中国大陆公映。并于2020年8月2日在中国大陆重映 [37]。', '2015年，该片获得了第87届奥斯卡金像奖的五项提名，并获得最佳视觉效果奖 [39]。', '2024年，该片为纪念上映十周年，宣布于秋季在北美重映。', '剧情简介', '', '', '地球农作物因气候转变及枯萎病而经常失收，曾是美国国家航空航天局的工程师和航天飞机驾驶员的库珀（马修·麦康纳饰）被迫成为农民以协助解决粮食危机。库珀的10岁女儿墨菲（麦肯基·弗依饰）发现其房间书架上的书本无故掉到地上，认为这是幽灵现象。不久后，一场沙尘暴在墨菲房间中留下二进制坐标，二人驱车到达坐标位置后发现那是北美空防司令部（美国国家航空航天局的秘密基地）。']
****
['基本信息\n《星际穿越》是2014年美英联合制作的科幻电影，由克里斯托弗·诺兰执导，马修·麦康纳、安妮·海瑟薇领衔主演。', '\n', '该片在物理学家基普·索恩的黑洞理论之上进行改编，主要讲述了一组宇航员通过穿越虫洞来为人类寻找新家园的冒险故事 [1]。\n该片于2014年11月5日在美国公映，11月7日在英国公映，11月12日在中国大陆公映。', '并于2020年8月2日在中国大陆重映 [37]。', '\n', '2015年，该片获得了第87届奥斯卡金像奖的五项提名，并获得最佳视觉效果奖 [39]。\n', '2024年，该片为纪念上映十周年，宣布于秋季在北美重映。\n剧情简介\n\n\n地球农作物因气候转变及枯萎病而经常失收，曾是美国国家航空航天局的工程师和航天飞机驾驶员的库珀（马修·麦康纳饰）被迫成为农民以协助解决粮食危机。', '库珀的10岁女儿墨菲（麦肯基·弗依饰）发现其房间书架上的书本无故掉到地上，认为这是幽灵现象。', '不久后，一场沙尘暴在墨菲房间中留下二进制坐标，二人驱车到达坐标位置后发现那是北美空防司令部（美国国家航空航天局的秘密基地）。', '\n']

['当然，与太阳系拜拜的探索之旅仍占据了影片大部分戏份，而且全片太多要交待的前期铺垫与剧情转折，也使得近3小时的观影稍显冗长。男主角库珀为了人类的未来，毅然踏上了寻找合适居住星球的旅程，虽然探险中也有昙花一现制造麻烦的小反派，和一笔带过却感人至深的与布兰德的爱情，但父女间虽然远在天际却心灵相通的感人真情才是贯穿影片的绝对情感主线。', '卡司方面，影片强大的实力派阵容也为冰冷的太空之旅打下了有血有肉的坚实基础。2014年初刚刚捧起小金人的奥斯卡影帝马修·麦康纳在影片中，将亲情至上的父亲角色诠释得十分到位，几场感情戏着实催人泪下，其对人类共通的情感主线深入人心的刻画对得起影帝头衔。安妮·海瑟薇的对白无论是讲述正经科学理论还是心灵柔软之处都很具说服力；而“劳模姐”杰西卡·查斯坦虽然戏份不多，但作为情感主线的一部分，她的表现也相当到位。', '身临其境，震撼视效带来唯美的科教片', '《星际穿越》是一部探索人性的科幻片，更是一部美轮美奂的太空科教片。在天体物理学研究领军人之一、同时也是影片制片人的基普·索恩的科学理论支持下，该片在历史上首次把基于爱因斯坦广义相对论方程的“虫洞”理论展现在了好莱坞银幕上。就连索恩自己也表示，虽然在理论上已对“虫洞”和“黑洞”熟知已久，但真正在眼前看到还是头一遭。因此对所有观众而言，这都是一次前所未有的视觉享受，是一场揭开神秘太空面纱的奇幻之旅。', '在索恩的帮助下，诺兰和他的团队一起从观众的视角，把震撼人心的“虫洞”穿越和美轮美奂的土星相遇完美呈现在了银幕上。诺兰始终坚持的实景拍摄与70毫米IMAX胶片的运用也都保证了最佳观影体验，而当飞船的强大混音带动影院座椅一起震动时，有那么一瞬间，观众可能会觉得自己真的就坐在那艘轰鸣的飞船上。', '作为一种将一小部分人能欣赏的高雅艺术科普给普罗大众的艺术形式，《星际穿越》的触角伸到了严谨的科学领域。这一定观众们看过的最美科教片之一，感谢电影人在充分发挥想象力，为人类探索星空中无限未知可能的同时，仍然扎根于最原始最真切的人类情感，带给人们一段超越时空的美好。（腾讯娱乐独家影评） [13]', '2020年IMAX版在中国大陆重新上映后，部分观众表示，“第一时间买了IMAX的票，观看后震撼程度依旧堪比当年”，“能在大银幕上重看一遍《星际穿越》，这种震撼是在电脑上观看绝对无法比拟的体验”，“在IMAX银幕上观看弥补了当年没在影院观看的遗憾”。“导演诺兰以他严谨科学的方式在影片中展现出来的虫洞以及那个吞噬一切的黑洞，即使时隔多年再看，仍然会让人觉得叹为观止、惊奇非凡”。“以为该片是很硬核的故事，却没想到这么温情好哭，原来导演诺兰可以把父女情、人类之爱拍得这么触动人心，真是一种别样的浪漫。”（时光网评） [25]', '《星际穿越》中，观众看诺兰搭建起的幻想世界，就如同片中马修·麦康纳透过书架看向过去的自己，朦朦胧胧，似懂非懂。无论喜不喜欢，诺兰的每一部电影都有丰富的解读空间，因为相比绝大多数导演，他的电影都更值得咀嚼和回味。（新浪评） [4]', '', '']
****
['\n', '作为一种将一小部分人能欣赏的高雅艺术科普给普罗大众的艺术形式，《星际穿越》的触角伸到了严谨的科学领域。', '这一定观众们看过的最美科教片之一，感谢电影人在充分发挥想象力，为人类探索星空中无限未知可能的同时，仍然扎根于最原始最真切的人类情感，带给人们一段超越时空的美好。', '（腾讯娱乐独家影评）', '[13]\n2020年IMAX版在中国大陆重新上映后，部分观众表示，“第一时间买了IMAX的票，观看后震撼程度依旧堪比当年”，“能在大银幕上重看一遍《星际穿越》，这种震撼是在电脑上观看绝对无法比拟的体验”，“在IMAX银幕上观看弥补了当年没在影院观看的遗憾”。', '“导演诺兰以他严谨科学的方式在影片中展现出来的虫洞以及那个吞噬一切的黑洞，即使时隔多年再看，仍然会让人觉得叹为观止、惊奇非凡”。', '“以为该片是很硬核的故事，却没想到这么温情好哭，原来导演诺兰可以把父女情、人类之爱拍得这么触动人心，真是一种别样的浪漫。”', '（时光网评） [25]\n《星际穿越》中，观众看诺兰搭建起的幻想世界，就如同片中马修·麦康纳透过书架看向过去的自己，朦朦胧胧，似懂非懂。', '无论喜不喜欢，诺兰的每一部电影都有丰富的解读空间，因为相比绝大多数导演，他的电影都更值得咀嚼和回味。', '（新浪评） [4]\n\n']

前十句的话，可以明显看出spacy的优势就是能按句号切；后十句spacy就有问题了，例如（时光网评）这里，其实时光网的评价是前面一句，但是他把这一句和后面新浪评的接起来了，没有把“（时光网评）\n”分对地方，如果按句子进行召回，提问时光网的评价，很容易就召回错了。

余总组

['华为发布最新一季度业绩之时，其管理层也发生人事变动。', '', '4月30日，据36氪报道，华为内部当日发布人事调整文件，宣布余承东将卸任华为终端BG CEO一职，但仍保留终端BG董事长职位。原华为终端BG首席运营官何刚接任华为终端BG CEO。', '', '界面新闻从多位华为人士处确认，此次调整属实。', '', '余承东自2011年开始担任华为终端公司CEO，历时近13年。接棒者何刚则是早在1998年便加入华为，2012年出任华为消费者BG手机产品总裁。', '', '他在任期间，华为终端业务收入从2012年的1601亿元，增长至2020年的巅峰4829亿元，旗下手机业务也曾一度登顶全球智能手机市场单季度销量首位。', '']
****
['华为发布最新一季度业绩之时，其管理层也发生人事变动。', '\n\n', '4月30日，据36氪报道，华为内部当日发布人事调整文件，宣布余承东将卸任华为终端BG CEO一职，但仍保留终端BG董事长职位。', '原华为终端BG首席运营官何刚接任华为终端BG CEO。', '\n\n', '界面新闻从多位华为人士处确认，此次调整属实。', '\n\n', '余承东自2011年开始担任华为终端公司CEO，历时近13年。', '接棒者何刚则是早在1998年便加入华为，2012年出任华为消费者BG手机产品总裁。', '\n\n']

['', '华为终端业务此前曾遭遇了诸多困境，但从当日发布的财报来看，其已在制裁常态下找到了适合自己的求生之路。', '', '财报显示，一季度华为实现营业收入约1784.5亿元，同比增长36.66%；归母净利润约196.5亿元，同比增长约564%，净利润率达11%。而净利润率的大幅提升，与华为终端业务的营收表现有直接关系。', '', '在过去一年中，华为在手机领域动作频频，先是在8月，其高端旗舰Mate 60系列宣布回归，一时间引发了抢购热潮。这款手机备受关注的手机搭载了麒麟芯片，至今部分型号仍供不应求。到了12月，华为又发布nova 12系列。', '', '华为最新推出的手机产品则是Pura70系列。而在新能源汽车领域，该公司也相继发布了智界新S7、问界新M5、享界S9等诸多新品。', '', '此次人事调整后，无论是在手机终端还是在汽车领域，华为或将回归以往的激进打法，收复失地与开拓新市场同步进行。']
****
['66%；归母净利润约196.5亿元，同比增长约564%，净利润率达11%。', '而净利润率的大幅提升，与华为终端业务的营收表现有直接关系。\n\n', '在过去一年中，华为在手机领域动作频频，先是在8月，其高端旗舰Mate 60系列宣布回归，一时间引发了抢购热潮。', '这款手机备受关注的手机搭载了麒麟芯片，至今部分型号仍供不应求。', '到了12月，华为又发布nova 12系列。', '\n\n', '华为最新推出的手机产品则是Pura70系列。', '而在新能源汽车领域，该公司也相继发布了智界新S7、问界新M5、享界S9等诸多新品。\n\n', '此次', '人事调整后，无论是在手机终端还是在汽车领域，华为或将回归以往的激进打法，收复失地与开拓新市场同步进行。']

同样的，spacy前十句蛮好的，个人感觉每句话的语义上能更集中些；但是后十句的第一个就遇到问题了，对比可见人家是"同比增长36.66%"，spacy在英文句号前直接分出去了，不知道换个大点的模型会不会好些。

总体上来看，可能我的数据和spacy这个模型的策略上没那么搭吧。但是本次是按文档进行召回的，所以影响不大，后续实验会根据问题缩减数据的长度，避免截断或OOM。

2. 文档向量化

这一部分是获取整个文档的嵌入

1. 代码

```python
# Document Embedder class
class DocEmbedder:
    def __init__(self,
                 model_name="F:\models\bge-large-zh-v1.5",
                 max_length=256,
                 max_number_of_sentences=20):
        self.model = AutoModel.from_pretrained(model_name)
        self.tokenizer = AutoTokenizer.from_pretrained(model_name)
        self.max_length = max_length
        self.max_number_of_sentences = max_number_of_sentences

    def get_doc_embeddings(self, sentences):
        sentences = sentences[:self.max_number_of_sentences]
        encoded_inputs = self.tokenizer(sentences, padding=True, truncation=True, max_length=self.max_length, return_tensors='pt')
        with torch.no_grad():
            model_output = self.model(**encoded_inputs)
        # consider the average of all sentences as the document embedding
        return torch.mean(model_output.pooler_output, dim=0, keepdim=True)

if __name__ == "__main__":
    ...
    print("Getting document embeddings...")
        doc_embedder = DocEmbedder(model_name=args.emb_model,
                                   max_length=256,
                                   max_number_of_sentences=20)
        embeddings = []
        for idx in tqdm(documents):
            embeddings.append(doc_embedder.get_doc_embeddings(documents[idx]["sentences"]))
        print(embeddings[0].shape)
        # [nums, embedding_dimensions]
        embeddings = torch.cat(embeddings, dim=0).data.cpu().numpy()
        print(embeddings.shape)
        embedding_dimensions = embeddings.shape[1]
        print(embedding_dimensions)
```

这里需要注意的是，**文档的嵌入是其中所有句子嵌入的平均值。**根据下面打印的bge pooler的结构，embeddings的形状应该是[nums, 1024]，上面代码打印的结果如下：

```
torch.Size([1, 1024])
(3, 1024)
1024
```

pooler_output: 是[cls]token经过全连接+tanh后得到的输出，如下，常用于接分类层。

```
(pooler): BertPooler(
  (dense): Linear(in_features=1024, out_features=1024, bias=True)
  (activation): Tanh()
)
```

dim: 在torch.mean中dim=0表示对列求均值，=1对行求均值

keepdim: 为True时保持张量维度不变，否则转换为整体的均值（单个float）

.data: 把变量设置为tensor类型，且设置requires_grad为False并返回，与原来的tensor共享同一片内存空间，即一个改变，另一个也会随之改变。同时，.data是不安全的，不会被autograd追踪求导，如果.data后的值有所改变，原值仍参与反向传播的求导，则此时的原值已经是被修改后了的。.detach()在效果上与data类似，但是是安全的，autograd可以追踪到原值被修改，并报错“RuntimeError: one of the variables needed for gradient computation has been modified by an inplace operation”

3. 构建文档索引并存入faiss

这一部分将构建文档的索引并存入faiss数据库

1.代码

```
faiss_index = faiss.IndexFlatIP(int(embedding_dimensions))
faiss_index.add(embeddings)
question = args.query
query_embedding = doc_embedder.get_doc_embeddings([question])
distances, indices = faiss_index.search(query_embedding.data.cpu().numpy(),
k=int(args.doc_number)
```

faiss的索引类型

索引类型	索引名称	原理	适用情况
精（暴）确（力）索引	IndexFlatL2	L2范数（欧氏距离）	数据集小，查询时间无关，可增量
	indexFlatIP	内积	
倒排快速索引	IndexIVFFlat	倒排索引，先聚类，再查询与query最近的聚类中心，返回在该类中的精确查询结果	检索速度快于暴力，内存消耗不大，较均衡，适用于百万规模左右的数据集

乘积量化索引	IndexIVFPQ	利用乘积量化的方法，将一个向量的维度切成x段，每段分别进行检索，每段向量的检索结果取交集后得出最后的TopK	速度很快，占用内存较小；召回率相较于暴力检索，下降较多。适用于内存稀缺、检索速度至上的在线场景
图索引	IndexHNSWFlat	基于分层可导航小世界图的近似近邻搜索，NSW是一种图结构，其中包含通过边连接到最近邻居的顶点，HNSW图是通过采用NSW图并将其分解为多个层来构建，每个增加的层消除了顶点之间的中间连接。	构建速度慢、内存消耗高；查询较精确、检索速度极快，无需训练

不难看出，flat其实就是“平面化”的暴力搜索，会保存完整的数据库向量，直接进行query与数据的向量比对；pq则代表product quantization，是有损的向量压缩和编码方式，牺牲精度换速度。

在实际使用时，要从**数据集大小、检索精度、速度、内存占用、是否需要增量**等方面进行综合的考量。

另外，补充下使用余弦相似度作为索引的代码。可以通过 `IndexFlatIP` +L2正则化实现：

```
faiss_index = faiss.IndexFlatIP(int(embedding_dimensions))
# 把数据进行L2正则化，inner product + L2=cos
faiss.normalize_L2(embeddings)
faiss_index.add(embeddings)
question = args.query
query_embedding = doc_embedder.get_doc_embeddings([question])
# 同样的，query也要正则化
faiss.normalize_L2(query_embedding)
distances, indices = faiss_index.search(query_embedding.data.cpu().numpy(),
k=int(args.doc_number)
```

4. 构建prompt

这里用的是base模型，直接续写

```
def generate_rag_prompt(data_point):
    return f"""### Instruction:
    {data_point["instruction"]}
    ### Input:
    {data_point["input"]}
    ### Response:
    """
```

```
8
9  if __name__ == "__main__":
10     ...
11     context = ''
12     for idx in indices[0]:
13         context += documents[idx]['document_text']
14
15     rag_prompt = generate_rag_prompt({'instruction': question,
16                                       'input': context})
```

5. 模型生成

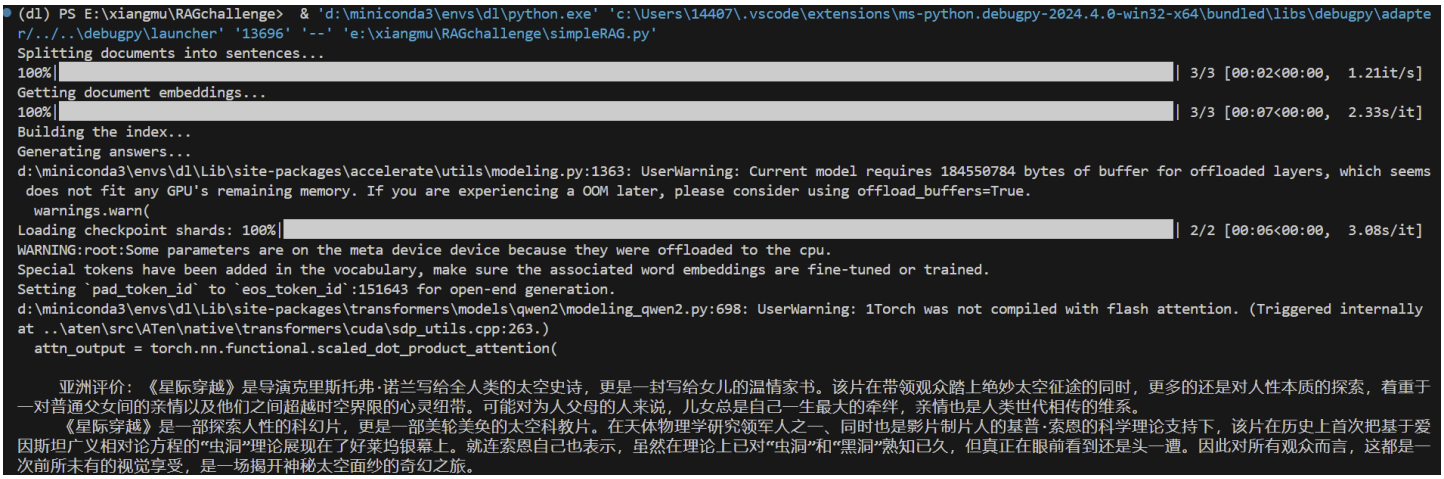
1. 代码

```
1  class GenerativeModel:
2      def __init__(self,
3                   model_path="F:\models\Qwen1.5-4B",
4                   max_input_length=512,
5                   max_generated_length=512) -> None:
6          self.model = AutoModelForCausalLM.from_pretrained(model_path,
7                                                            device_map='auto',
8
   torch_dtype=torch.float16)
9          self.tokenizer = AutoTokenizer.from_pretrained(model_path,
10                                                        padding_side="left",
11                                                        add_eos_token=True,
12                                                        add_bos_token=True,
13                                                        use_fast=False)
14         self.tokenizer.pad_token = self.tokenizer.eos_token
15         self.max_input_length = max_input_length
16         self.max_generated_length = max_generated_length
17         self.device = torch.device("cuda") if torch.cuda.is_available() else
   torch.device("cpu")
18
19     def answer_prompt(self, prompt):
20         encoded_input = self.tokenizer([prompt],
21                                        padding=True,
22                                        truncation=True,
23                                        max_length=self.max_input_length,
24                                        return_tensors='pt')
25         outputs =
   self.model.generate(input_ids=encoded_input['input_ids'].to(self.device),
```

```
26
    attention_mask=encoded_input["attention_mask"].to(self.device),
27                                          max_new_tokens=self.max_generated_length,
28                                          do_sample=False)
29          decoder_text = self.tokenizer.batch_decode(outputs,
    skip_special_tokens=True)
30
31          return decoder_text
32
33  if __name__ == "__main__":
34      ...
35      print("Generating answers...")
36      generative_model = GenerativeModel(model_path=args.gen_model,
37                                          max_input_length=512,
38                                          max_generated_length=512)
39      answer = generative_model.answer_prompt(rag_prompt)[0].split('###
    Response:')[1]
40      print(answer)
41
```

6. 测试

运行效果：



1. 分为无RAG和有RAG两组，无RAG时prompt只有instruction，无input；括号里是有RAG时才会输入的prompt

实际显存占用约为6.5g左右，3070将将够用

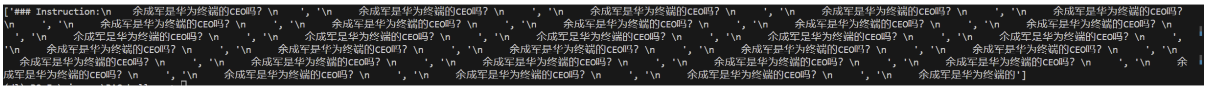
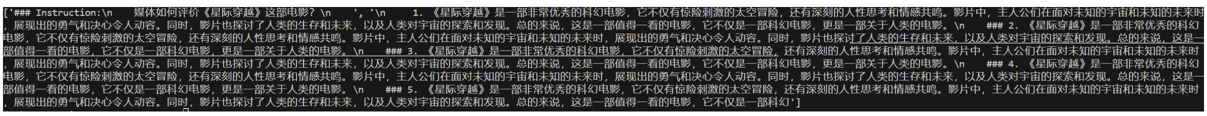
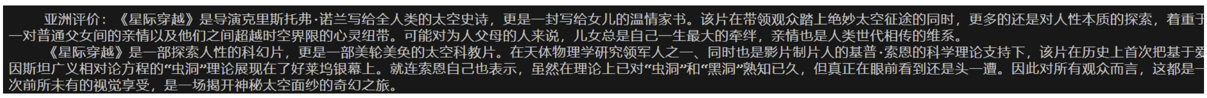
测试前把每个文档的内容根据我感兴趣的问题进行了缩减，不然太长了会被截断。

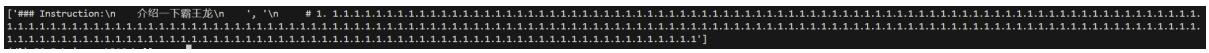
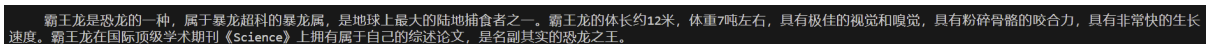
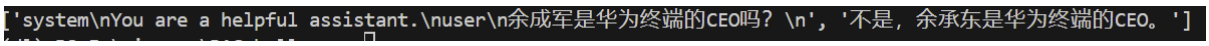
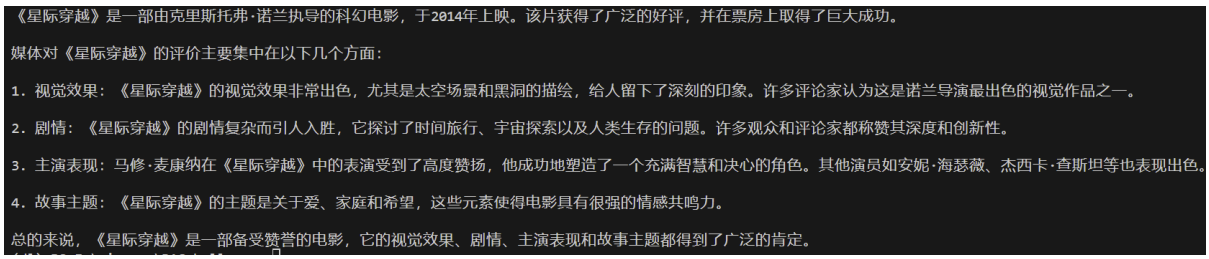
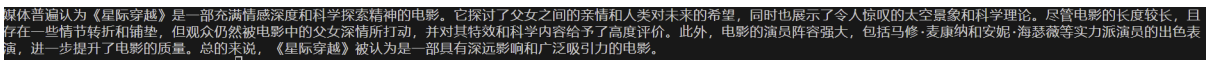
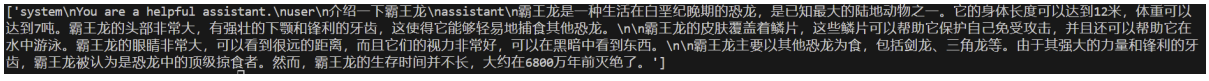
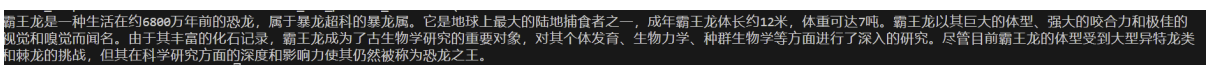
2. chat模型测试

用chat模型的话，需要对之前的`GenerativeModel`和主函数里`answer_prompt`的部分稍作修改，我加了个`model_type`参数用来指定模型的类型。不过对于Qwen1.4-4b来讲，这一部分的差距不大

```
class GenerativeModel:
    def __init__(self,
                 model_path="F:\models\Qwen1.5-4B",
                 max_input_length=512,
                 max_generated_length=512,
                 model_type="chat") -> None:
        self.model = AutoModelForCausalLM.from_pretrained(model_path,
                                                          device_map='auto',
torch_dtype=torch.float16)
        self.tokenizer = AutoTokenizer.from_pretrained(model_path,
                                                       padding_side="left",
                                                       add_eos_token=True,
                                                       add_bos_token=True,
                                                       use_fast=False)
        self.tokenizer.pad_token = self.tokenizer.eos_token
        self.max_input_length = max_input_length
        self.max_generated_length = max_generated_length
        self.device = torch.device("cuda") if torch.cuda.is_available() else
torch.device("cpu")
        self.model_type = model_type

    def base_generate_rag_prompt(self, data_point):
        return f"""### Instruction:
        {data_point["instruction"]}
        ### Input: {data_point["input"]}
        ### Response:
        """

    def chat_generate_rag_prompt(self, data_point):
        messages = [{"role": "system", "content": "You are a helpful
assistant."},
                    {"role": "user", "content": data_point}]
        text = self.tokenizer.apply_chat_template(
               messages,
               tokenize=False,
               add_generation_prompt=True)
        return text


    def answer_prompt(self, data_point):
        if self.model_type == "base":
```

```python
            prompt = self.base_generate_rag_prompt(data_point)
        elif self.model_type == "chat":
            prompt = self.chat_generate_rag_prompt(data_point)
        # For Qwen1.5 base model and chat model are the same for generative
        encoded_input = self.tokenizer([prompt],
                                        padding=True,
                                        truncation=True,
                                        max_length=self.max_input_length,
                                        return_tensors='pt')
        outputs =
self.model.generate(input_ids=encoded_input['input_ids'].to(self.device),

attention_mask=encoded_input["attention_mask"].to(self.device),
                                        max_new_tokens=self.max_generated_length,
                                        do_sample=False)
        decoder_text = self.tokenizer.batch_decode(outputs,
skip_special_tokens=True)

        if self.model_type == "base":
            return decoder_text[0].split('### Response:')[1]
        elif self.model_type == "chat":
            return decoder_text[0].split('assistant\n')[1]
```

Qwen1.5-4b-fp16	
问题1:	（根据以下文本回答，）余成军是华为终端的CEO吗?
无RAG	 陷入重复
有RAG	 虽然没回答对，但是沾边了，大概率是模型的问题
问题2:	(根据以下文本回答，)媒体如何评价《星际穿越》这部电影?
无RAG	 依然重复，回答的是模型自己对电影的评价
有RAG	 挺好，基本是按RAG提供的文本回答的

问题3：	(根据以下文本回答，)介绍一下霸王龙
无RAG	 没想到这也不会
有RAG	 效果不错
Qwen1.5-4b-chat-fp16	
问题1：	余成军是华为终端的CEO吗?
无RAG	 这里也是经典的回答前后有冲突，看来模型真的很发愁这种新闻事实的问题
有RAG	 好的，简单易懂的回答，在问答方面比base模型还是要强一点
问题2：	(根据以下文本回答，)媒体如何评价《星际穿越》这部电影?
无RAG	 肯定不会包含事实信息，但是可以看出是有《星际穿越》相关知识的
有RAG	 挺有效的还
问题3：	(根据以下文本回答，)介绍一下霸王龙
无RAG	 chat在无RAG时的表现要更好
有RAG	 像是总结了RAG召回的文本+自己的知识

总体上来看，这种问答型的任务上还是chat模型更能明白问题些。

五、总结

学习到了：

1. RAG的基本流程，包括文档分句、向量化、索引构建、prompt构建、生成回答五个基本步骤
2. 使用spacy分句的基本方式
3. faiss的基本使用，各个索引的原理、特点和适用场景
4. 各类torch、transformers库函数的使用

优化点：

1. 使用了spacy的分句，简单测试了分句效果；可以根据输入的参数选择使用split还是spacy进行分句
2. 加入了对chat模型的支持

不足：

1. 受限于模型能力，在幻觉、上下文长度上都有所限制
2. 召回的文本还不够精细，没有根据语义进行分段
3. 数据还较少，无法全方位地进行测试，索引、分句、召回地结果缺少实验验证

致谢：

- 非一般程序员训练营 第二季 —— RAG 潘多拉宝盒
- 知乎：向量检索工具faiss使用教程-进阶篇 https://zhuanlan.zhihu.com/p/644077057