

document-question-answering-bot

（文档问答机器人）

项目文档

队伍名称：ArkShin

项目成员：邱昱力（北京理工大学计算机学院）

钱星雨（北京理工大学计算机学院）

何婉欣（北京理工大学计算机学院）

指导教师：郭宇航（北京理工大学计算机学院）

吴荣杰（统信软件）

一、赛题描述

Deepin wiki 上有 900 多条 deepin 系统相关的中文教程和词条，请编写能根据这些内容回答问题的聊天机器人。使用者通过命令行界面输入问题，机器人输出回答和参考的 wiki 文档的链接。

聊天机器人要能根据 deepin wiki 内容回答问题；且回答和问题有 80%以上概率的相关性；对于每个问题，机器人要给出 3 个可能的回答；编写博客，记录开发过程的心得与体会。

二、项目完成情况

如下表，本项目共有 3 项必须任务和 2 项可选任务。我们在全部完成之后与出题老师进行沟通，又添加了两项自选任务。

| 任务目标 | 任务类型 | 完成情况 |
|--|------|----------------------------|
| 完成聊天机器人，能根据 deepin wiki 回答问题 | 必须 | 完成的机器人能根据 deepin 回答问题 |
| 训练模型，回答和问题有 80%以上概率的相关性 | 必须 | 回答准确性达到 90%以上 |
| 编写博客记录开发过程的心得体会 | 必须 | 完成 |
| 添加 https://linglong.dev/ 使用手册内容，支持回答玲珑问题 | 可选 | 完成 |
| 使用 web 或 qt 为机器人添加可视化界面支持，能在 deepin 系统使用 | 可选 | 完成，添加了 web 可视化支持 |
| 机器人支持英汉双语问答 | 自行添加 | 完成 |
| 机器人提供多样化问答能力 | 自行添加 | 完成，信息不匹配时机器人会输出“我不知道”或发起追问 |

三、题目分析与设计思路

这道题的本质是制作一个问答系统，是 NLP 领域的任务。

3.1 相关工作概述

问答系统在传统上存在以下技术路线。一种是利用知识图谱构建 deepin 系统各概念之间的关系，这需要我们从 900 多个网页中抽取实体并构建关系。但这需要较大的人工投入，针对知识图谱进行问答也不够自然。

以 BERT 为代表的预训练语言模型 (Pretraining Language Model, 以下简称 PLM) 代表了第二类实现方法。这需要我们用 deepin wiki 对 BERT 等 PLM 模型进行微调 (fine tune); 并将问题作为输入进行微调。这类方法可以完成一些简单的问答任务，但需要一定算力投入。

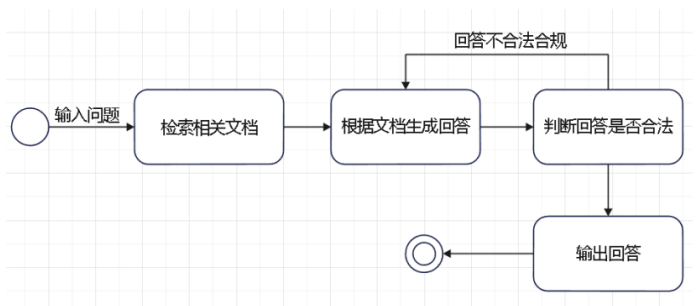
以 ChatGPT 为代表的大语言模型 (Large Language Model, 以下简称 LLM) 为问答系统的实现带来了新的思路。模型参数的增大极大提升了问答系统的表现，可以利用模型自身数据中的“固有知识”直接构建问答系统。但这类 LLM 存在幻觉问题，直接构建无法保证输出的可靠性。鉴于 LLM 巨大的参数规模和未开源性，像前述的 PLM 一样进行微调变得难以实现。

使用 LLM 结合检索系统构建新的问答系统是问答系统的新实现形式。OpenAI 基于此路线利用必应搜索得到参考网址并输入 GPT3 作答及微调，搭建了 WebGPT[1]。

3.2 系统框架

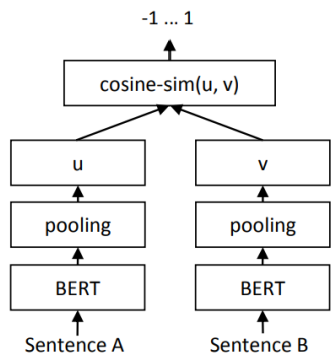
参考前述 WebGPT 的思路，我们的问答系统使用检索系统和 LLM 的形式构成。Deepin wiki 和 linglong.dev[2]上的近千个文档构成了我们的系统外部数据库。针对用户的输入问题，系统先从外部数据库中检索出最相关的文档内容；再将文档内容和问题一同作为 prompt 输入 LLM，使之参考生成答案。这么做既结合了 LLM 回答流畅自然的优点，还用外部知识库避免了幻觉问题，确保了输出的真实可靠。

该问答系统的状态图如下：



3.3 检索系统的设计思路

我们使用 sBERT[3]这一双塔模型 (Deep Structured Semantic Model, DSSM) 实现检索系统。双塔模型包括 user 塔和 item 塔；item 塔负责对所有文档进行词嵌入得到文档的向量表示。user 塔负责对输入的问题进行词嵌入得到语句的句向量。计算向量的余弦距离可得到推荐的结果。下图是 sBERT 的示意：



3.4 回答生成的设计思路

前述的检索系统会返回相关度最高的 3 个参考文档。这里将问题和参考文档一起作为 prompt 逐个传给 LLM 并使之回答问题。在这里我们使用的 LLM 包括 gpt-3.5，百度的 ErnieBot 和讯飞的 Spark。用户可根据需要从中选择任意模型进行提问。

同时我们对 LLM 的 prompt 进行了修改，从而实现了多样化问答能力。当文档和问题相关时，直接输出答案；当文档和问题无关时，输出“对不起，我没有找到相关内容”；当问题不够详细时，LLM 会主动对用户进行追问。受限于基础 LLM 的性能，多样化问答功能仅在 gpt-3.5 上得到实现。

四、实现重点描述

词嵌入(word-embedding)即对词语进行向量化表示，使向量能反映语义。比对多种嵌入方式，我们最终使用 BERT 模型对文档进行词嵌入，得到了词向量的合理表示。在这里我们试验了不同的预训练 BERT，根据表现最后选择 Chinese-BERT-wwm[4] 预训练模型。

BERT 只是对每个词进行了嵌入，之后还需要通过池化(mean-pooling 或 max-pooling)得到整个文档或句子的句向量，这里我们使用 mean-pooling 完成池化。将用户问题的嵌入向量（user 塔）和所有文档的嵌入向量（item 塔）逐一比较余弦距离，距离最小的相似度最高。

Chinese-BERT-wwm 模型只在中文上得到了较好的嵌入效果。对

于输入为英文的情况，问答系统会先利用 LLM 将输入翻译为中文再进行词嵌入；输出时会指定 LLM 用英文输出。

为了加快系统的检索速度，我们把所有文档嵌入后得到的向量都存储在对应的 .json 文件中(这在仓库的 embedding 目录下)。当给定问题类型后会直接比较各向量间的距离，而不用再对文档进行嵌入。

调用 LLM 的 API 进行问答的过程中，对 LLM 的提示角色内容是：

"role": "system", "content": "你是一个自动问答系统；请按要求回答问题"

"role": "user", "content": "请根据<>中参考文档的内容回答问题，回答的语言要和提问的语言一致。你的回答中不必包括参考链接。请先从下面的 3 种情况中选择符合的情况，按要求输出：

当能确定答案时，直接输出答案；

当感到问题描述不够清晰时发起追问，输出格式是"追问：你的追问内容"；

当文档中没有匹配的内容时，输出"对不起，我没有从下面的文档中找到相关内容"；

问题：<{question}> 参考文档：<{ref}>"

如果文档过长超过了 LLM 的 token 限制，会对文档截断使之仅保留前面的内容；直到输出 3 个回答。每个回答之后系统会输出该文档所在的网页地址作为参考链接，这有助于使用者查看原文档，检验回答的正确性。

五、系统运行、测试及评价

5.1 系统测试

开发过程中针对每个函数体均进行了单元测试；针对文档匹配、LLM 提问等模块均进行了构件测试，确保项目各构件间的交互正确。

5.2 系统运行示意

我们的系统可以用 cmd 启动执行，也可以用 web 进行访问；便于在不同平台间进行迁移。系统的运行配置过程详见 readme.md 文档，更多的效果展示详见 demo.pdf。下图只作示意：



5.3 系统效果评价

为了衡量文档问答机器人的回答效果，我们从 deepin 文档中抽取信息构建了 30 个问-答对序列作为数据集存储在 dataset.json 中。

```
dataset.json
C:\Users\lenovo\Desktop>deepin QA> dataset.json
1 {
2   "A-Aquestion1": "介绍一下deepin的发展历史",
3   "answer1": "深度操作系统的历史可以追溯到 2004 年, 其前身 Hiweed Linux 是中国第一个基于 Debian 的本地化版本",
4   "A-Aquestion2": "deepin社区有哪些交流方式",
5   "answer2": "Deepin社区有多种交流方式可供选择: 论坛、通过Deepin官方论坛可以进行产品需求和问题反馈的讨论。讨论",
6   "A-Aquestion3": "deepin社区例会的目的是什么",
7   "answer3": "鼓励社区成员主动分享自身的近期工作, 促进社区成员之间的信息传递。对于想了解 deepin 社区的爱好者而",
8   "A-Bquestion4": "deepin的控制中心是什么? 有哪些功能?",
9   "answer4": "深度操作系统通过控制中心来管理系统的基本设置, 包括小工具介绍、常用设置以及账户管理、网络设置、日期",
10  "A-Bquestion5": "deepin如何进行恢复和备份",
11  "answer5": "Deepin操作系统的备份和恢复过程如下: 备份: 进入Deepin控制中心, 找到备份/还原选项。选择备份路径和备",
12  "A-Bquestion6": "我应该怎么用deepin上网?",
13  "answer6": "深度操作系统预装了chrome浏览器, 您可以打开浏览器浏览网页, 也可以在商店中下载其他浏览器访问因特网。",
14  "Bquestion7": "Linux内核错误是什么?",
15  "answer7": "内核错误(Kernel panic)是指操作系统在监测到内部的致命错误, 并无法安全处理此错误时采取的动作。这个",
16  "Bquestion8": "介绍一下微内核?",
17  "answer8": "微内核只管理它必须管理的东西: CPU、内存和IPC。计算机中几乎所有的东西都可以被看作是一个附件, 并且可",
18  "Bquestion9": "我不小心删除了文件, 我应该怎么恢复啊?",
19  "answer9": "关闭电源: 如果发现文件被删除, 立即按电源键关闭电脑, 避免关机过程中的数据写入操作进一步破坏丢失的",
20  "Bquestion10": "有什么快速清除文件的方法吗?",
21  "answer10": "有几种快速清空文件的方法。可以使用以下任意一种命令: $ > access.log。这个命令使用重定向操作符将",
22  "Cquestion11": "怎么获得wiki编辑权限?",
23  "answer11": "填写QQ文档表格, 获取编辑权限, 两天为一周期进行发放, 或者论坛发出请求, 后台维护人员主动赋权",
24  "Cquestion12": "软件引入是什么意思?",
25  "answer12": "一个软件的引入指的是一个软件(项目)申请被引入到社区某一个项目中, 依照本文描述的规则讨论, 进而满足",
26  "Cquestion13": "freedesktop是什么?",
27  "answer13": "freedesktop是一个开源组织。专注于开源图形和桌面系统的互操作性和共享技术。许多linux桌面环境都是
```

运行时每个问题系统都会给出最可能的 3 个答案，这样一共得到了 90 个结果。比较这些生成的答案和数据集参考答案之间的余弦相似度和 ROUGE-L 分数，以衡量其性能。

余弦相似度的计算公式为：

$Similarity = CosineDistance(RefAns.encode(), PredictAns.encode())$

即对参考答案和输出答案分别进行词嵌入得到句向量表示,比较句向量间的余弦距离。不同模型给出的每个回答的余弦相似度如下:

| | Ans1 | Ans2 | Ans3 |
|----------|---------------|---------------|---------------|
| GPT | 0.9282 | 0.9164 | 0.9135 |
| ErnieBot | 0.9022 | 0.9039 | 0.8748 |
| Spark | 0.9234 | 0.9212 | 0.9185 |

我们还选取 ROUGE-L 的准确率来反映输出答案的信息密度,其计算公式为:

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

其中 X,Y 分别表示参考答案和生成的答案; LCS 表示最长公共子序列(longest common subsequence)。

各模型整体的余弦相似度和 ROUGE-L 分数如下:

| | GPT-3.5 | ErnieBot | Spark |
|---------|---------|----------|--------------|
| 余弦相似度 | 0.919 | 0.894 | 0.921 |
| ROUGE-L | 0.33 | 0.281 | 0.455 |

鉴于余弦相似度利用 BERT 更多考虑了语义因素,相比 ROUGE-L 得分我们更倾向于使用该指标来评价模型效果。总的来看,各模型的准确率均达到 90%以上,满足项目要求;讯飞星火的 Spark 模型效果相对更好。系统生成的越靠前的答案准确率越高,这取决于推荐系统的推荐顺序;这也反映出推荐系统的有效性。

六、主要问题及解决方案

问题 1：如何搭建一个匹配效果良好的文档推荐系统？

解决方案：我们的思路源自 sentenceBERT 的论文，使用 BERT 进行嵌入并 pooling 后比较余弦相似度，能够较好对文档语义进行建模；其效果要优于直接把文档和问题都输入给 BERT 让其预测相似度的做法。

问题 2：怎么选择合适的词嵌入方式？

解决方案：词嵌入方式有 TF-IDF, Word2Vec, BERT 等，我们选择 BERT 进行词嵌入。考虑到处理文档内容是中文，我们在众多 BERT 预训练模型中选择了开源的 Chinese-BERT-wwm 模型，它以词语为单位进行嵌入，其表现优于 Google 的中文预训练 BERT。当然用户也可根据个性化需要更换其他预训练模型完成词嵌入，如使用 distiluse-base-multilingual-cased 来支持多语言输入。但根据我们的试验结果，Chinese-BERT-wwm 在中文领域的嵌入表现更佳，文档匹配效果更好。

问题 3：得到词向量之后该使用 mean-pooling 还是 max-pooling 得到句向量？

解决方案：根据 sentence transformer 上的各模型效果数据，选择 mean-pooling 进行操作。

问题 5：怎么构建数据集？

解决方案：数据集构建遵循了以下原则：一是覆盖全面，利用分层抽样确保各类问题均有涉及；二是利用现有 LLM(chatGPT 和文心一言) 辅助数据集构建；三是人工评判，对模型给出的输出进行审核以保证正确。

问题 6: LLM 有输入的 token 限制，参考文档太长怎么办？

解决方案: 一是对教程文档进行检查，过长的文档会被拆分成多个；二是在程序中进行判断，如果匹配的文档长度超过了 LLM 的输入限制会进行截断。三是我们选择的三个 LLM 最大 token 数量均大于 10k，基本满足了长度要求。

问题 7: 如何选择合适的 LLM？

解决方案: 目前主要的 LLM 有 GPT、Claude、LLaMa、ChatGLM、ErnieBot 等，我们这里选择 GPT、ErnieBot 和 Spark 完成实验。三者均能满足实验要求，而 Spark 在中文问答领域的效果略好。

七、收获

在项目的开发过程中，我们采用推荐系统+LLM 的路线来实现中文聊天机器人。

在项目开发过程中，我们遇到了一些问题并找到了相应的解决方案。例如，我们使用 BERT 模型进行词嵌入，并选择了 Chinese-BERT-wwm 模型，因为它在处理中文文档时表现更好。我们还使用了 mean-pooling 来得到文档或句子的句向量。为了评估模型的效果，我们构建了一个数据集，并通过计算余弦相似度和 ROUGE-L 来评估生成的回答的准确性。

这个项目最终呈现出良好的效果。技术上它解决了传统 QA 系统回答不够流畅的问题，也解决了生成式模型“胡说八道”的问题。sBERT 让机器人能够匹配出相似度高的文档，并在测试集上显示出这

个聊天机器人回答的高准确性；而大语言模型的应用在增强模型问答效果的同时也降低了复杂度，便于泛化推广。每个问题机器人都给出了三个可能的答案及参考链接，便于用户对信息的准确性进行核实。

在团队合作方面，我与我的团队成员密切合作，负责整合各个模块并进行最终的系统测试。我们共同努力，克服了项目中的各种问题，并成功完成了一个高效的中文聊天机器人。这个项目使我深入了解了 NLP 任务和相关技术，并提升了我的系统集成和测试能力。

八、致谢

感谢学校的郭宇航老师及统信软件的项目导师吴荣杰为我们提供了宝贵的思路和技术指导，帮我们明确了项目技术路线，对中途遇到的问题提出解决方案。最后感谢 NLP 领域的前辈研究者们，他们积累的学术成就让我们得以完成该项目。

参考文献：

- 1.WebGPT: Browser-assisted question-answering with human feedback, arXiv:2112.09332 [cs.CL]
- 2.Deepin wiki、linglong 使用手册
- 3.Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, arXiv:1908.10084 [cs.CL]
- 4.Revisiting Pre-Trained Models for Chinese Natural Language Processing, arXiv:2004.13922 [cs.CL]

人员分工：

邱昱力：统筹项目进展，完成 GPT 的调用模块编写；进行系统整合、优化、测试。

钱星雨：完成 sBERT 文档匹配部分的编写；编写博客并提交。

何婉欣：从 deepin 官网爬取全部文档；构建测试数据集；制作前端界面。