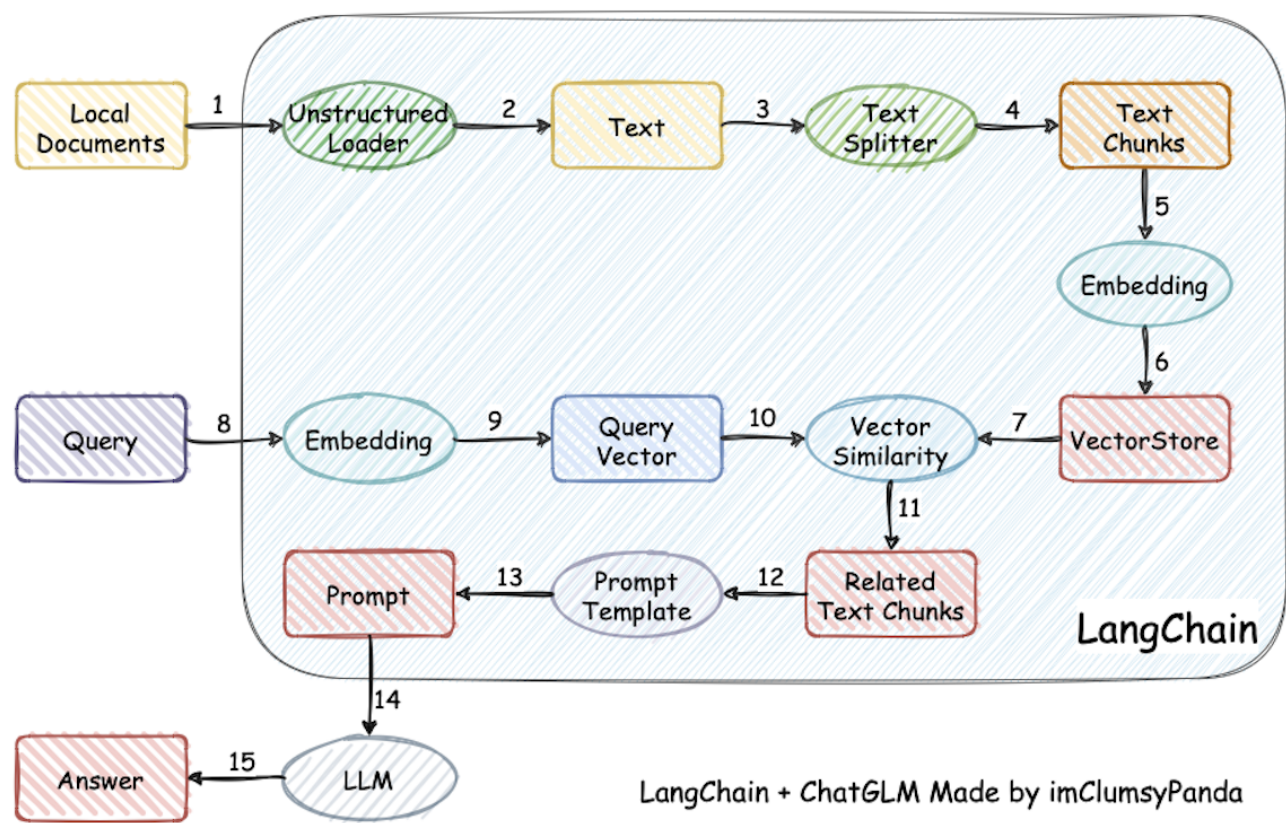


# 算法设计——基于langchain的本地知识库文本问答系统

## 架构图示



## 数据预处理部分

### 对于数据集的预处理

#### pdf to txt

使用langchain中的PyPDFLoader对于目录中的pdf文件进行提取,提取出文字, 写入到all\_data.txt里面。

做了去除'\n'的处理, 不好取出空格和全转换为小写的原因英文如果取掉空格就都成为连续的一段了 是有专有名词的大写, 比如"AI"等, 这些词在英文中是专有名词, 需要保留大写。

```
from langchain_community.document_loaders import PyPDFLoader
import os
files = os.listdir('data-test')
for file in files:
    loader = PyPDFLoader(f'data-test/{file}')
    docs = loader.load()
    with open('all_data.txt', 'a') as f:
        for doc in docs:
            doc.page_content = doc.page_content.replace('\n', '')
            f.write(doc.page_content)
```

## 提取有用部分

采取了一种“逆向索引”的方式，根据test\_A.csv里面question的内容来确定到底哪些信息是有关的。

如何准确确定有用信息满足题目要求且使得数据量尽可能小？我在这里提出来一种基于**罕见词**的信息抽取方法

想到这个思路的原因是

- 1.question的罕见词几乎都在文章中出现过
- 2.罕见词在全文中出现的数量较少，可以用较少量的信息完成任务

过程如下：

- 首先调用chatglm-6b，设定好messages,抽取question句子中的罕见词,返回一个句子中的两个罕见词

```
hanjian = []
def call_with_messages(string:str):
    #prompt = ["提取句子里面的2个比较长的罕见短语,越罕见越好,返回一个列表,格式为:
    ['罕见短语1', '罕见短语2'],不要加上其他任何内容"]
    messages = [
        {'role': 'system', 'content': "提取句子里面的2个罕见词,按照下面的格式返回一个列表:['罕见词1', '罕见词2']"},
        {'role': 'user', 'content': "提取句子里面的2个比较长的罕见词,越罕见越好,返回一个列表,格式为: ['罕见词1', '罕见词2'],不要加上其他任何内容, "+"句子为:"+string}]
    response = dashscope.Generation.call(
        'chatglm3-6b',
        messages=messages,
        # set the random seed, optional, default to 1234 if not set
        seed=random.randint(1, 10000),
        result_format='message', # set the result to be "message" format.
    )

    if response.status_code == HTTPStatus.OK:
        string = response.output['choices'][0]['message']['content']
        #print(string.strip('\n').strip(" "))
        string = string.strip('\n').strip(" ")
        print(string)
        list_items = eval(string)
        #print(list_items)
        # 有些词被大模型判断为罕见词，但其实在文章中并不罕见，我们需要把它去掉
        changyong = ['中兴', 'ZTE', "data", '中兴通讯', 'One', "ZTE公司", "算法", 'point cloud']
        for i in changyong:
            if i in list_items:
                list_items.remove(i)
        return list_items
    else:
        print('Request id: %s, Status code: %s, error code: %s, error message: %s' % (
```

```

        response.request_id, response.status_code,
        response.code, response.message
    ))
    return None

```

返回的罕见词表部分示例:

['混合五电平单相整流器', '四叉树数据结构', 'hybrid', 'quadtree', '非地面网络 (NTN)', '核心网络架构', 'non-terrestrial networks (NTN)', 'core network architecture', '载波点亮方案', '铁路通信网络业务潮汐效应', 'carrier lighting scheme', 'incoming vehicle recognition', '模型驱动', '智能简化通信系统', 'model-driven', 'intelligent simplified communication system', '智能维护模型', '通信网络', 'Intelligent Maintenance Model', 'communication networks', '5G PowerPilot', '节能目标', 'In the pursuit', 'enhancing network efficiency', 'ASTEO', 'Neutra', 'One', 'issuance', '5G-A时代', '网络 流量', '5G-A era', 'hundred or even a thousand times increase', '智能内生网络 (IEN)', '6G 网络 架构', 'intelligent endogenous network (IEN)', 'self-perception', 'self-analysis', 'self-decision', 'self-execution', 'self-assessment', 'UE', '5G LAN 技术', 'UE', 'Ethernet frames', '高铁网络规划', '覆盖规划误差', 'high-speed rail network planning', 'digital twin technology', '计算

再把文章拆分成CHUNK\_SIZE = 200, OVERLAP\_SIZE = 0的小段, 如果一个chunk包含了这些单词把它认为是有用的段, 并把它之前和之后的一段也加入以保证语义的连续性。这种分法相较于直接按照chunk和overlap进行切分, 使得关键词会处于相对在每个段落中心的位置上, 从而使得语义的连续性更高。

```

import random, os
from langchain_text_splitters import RecursiveCharacterTextSplitter
os.chdir('D:\Langchain-Chatcat\webui_pages\dialogue')
hanjian = lst
# f_tar = open('example.txt', 'r', encoding='utf-8')

```

```

text_splitter = RecursiveCharacterTextSplitter(
    # Set a really small chunk size, just to show.
    chunk_size=100,
    chunk_overlap=0,
    length_function=len,
    is_separator_regex=False,
)

```

```

with open('all_data.txt', 'r', encoding='utf-8') as f:
    lines = f.read()
    texts = text_splitter.create_documents([lines])
def filter_text_blocks(word_list, text_blocks):
    """

```



从给定的文本块中过滤出包含单词列表中任意一个单词的块。

参数:

word\_list (list): 包含需要检查的单词的列表

text\_blocks (list): 包含需要检查的文本块的列表

```
    返回：
    list: 过滤后的文本块列表
    """
    filtered_blocks = []
    for i in range(len(text_blocks)):
        for word in word_list:
            if word in text_blocks[i].page_content:
                for k in range(i-1,i+2):
                    if(k>=0 and k<len(text_blocks)):
                        if text_blocks[k].page_content not in filtered_blocks:
                            filtered_blocks.append(text_blocks[k].page_content)
                break
    return filtered_blocks
result = filter_text_blocks(hanjian, texts)
print(result)
with open('useful_data_100.txt', 'w',encoding='utf-8') as f:
    for line in result:
        f.write(line)
        #f.write('\n')
```

 all_data.txt	2024/6/26 20:47	文本文档	5,695 KB
 useful_data_100.txt	2024/6/30 1:26	文本文档	1,321 KB

创建知识库

可以看出，useful\_data内容远远小于all\_data内容，实现了比较好的有用内容提取。

把useful\_data.txt放进数据库路径Langchain-Chatcat\knowledge\_base\zte-100\content中，然后在langchain-chatcat的webui界面点击选择知识库管理->新建知识库->上传知识文件->添加文件到知识库->依据原文件重建向量库。这样就创建好了一个向量数据库

请输入知识库介绍:

关于zte-100的知识库

文件处理配置

单段文本最大长度: 400 - + 相邻文本重合长度: 100 - + ☐ 开启中文标题加强

添加文件到知识库

知识库 zte-100 中已有文件:

知识库中包含源文件与向量库，请从下表中选择文件后操作

序号	文档名称	文档加载器	分词器
1	useful_data_100.txt	UnstructuredFileLoader	ChineseRecursiveTextSplitter

1 to 1 of 1    < < Page 1 of 1 > >

query获取

因为一过分英文问题是从中文论文中提取的，一部分中文问题是从英文论文中提取的，我采取的策略是把问题分别翻译成中文和英文，再拼起来 从而使问题能和中文和英文论文匹配。前面获取罕见词的部分也需要从中文问题和英文问题中分别都提取以防缺漏。translate使用qwen2-7b-instruct进行翻译。

```
for i in range(len(data1)):
    trans = translate(data1[i])
    print(data1[i])
    print(trans)
    prompt.append(data1[i]+trans)
    hanjian += call_with_messages(data1[i])
```

```

    hanjian += call_with_messages(trans)
    print(hanjian)
df = pd.DataFrame(prompt)
print(df)
df.to_csv('questions.csv', index=False)

```

翻译模块:

```

import random
from http import HTTPStatus
import dashscope
from dashscope.api_entities.dashscope_response import Role

def translate(string:str):
    prompt = ["如果句子是中文则翻译成英文，如果句子是英文则翻译成中文，返回翻译结果"]
    messages = [
        {'role': Role.SYSTEM, 'content': "翻译句子（句子为英文则翻译为中文，句子为中文则翻译为英文），返回翻译结果"},
        {'role': 'user', 'content': string}]
    response = dashscope.Generation.call(
        'qwen2-7b-instruct',
        messages=messages,
        #prompt= prompt,
        # set the random seed, optional, default to 1234 if not set
        seed=random.randint(1, 10000),
        result_format='message', # set the result to be "message" format.
    )

    if response.status_code == HTTPStatus.OK:
        string = response.output['choices'][0]['message']['content']
        string = string.strip('\n')

        return string
    else:
        print('Request id: %s, Status code: %s, error code: %s, error message: %s'
% (
            response.request_id, response.status_code,
            response.code, response.message
        ))
        return None
if __name__ == '__main__':
    translate("The hybrid five-level single-phase rectifier proposed in the paper utilizes a quadtree data structure for efficient point cloud representation and compression.")

```

现在，每个问题都包括中文和英文了。

## 问答部分

先从questions.csv文件中把问题读取出来



```

result = []
import pandas as pd
data = pd.read_csv(r'webui_pages\dialogue\questions.csv', encoding='utf-8')
data1 = data.iloc[:,0]

for i in range(len(data1)):
    prompt = data1[i]

```

采用langchain-chatchat进行知识库问答

- 选择模型为qwen1.5-14b-chat,
- 选定对话历史记录条数为0, 因为每个问题是互相独立的, 不存在上下文依赖关系。
- 知识库匹配条数选择为5条, 因为我发现采用默认的3条时会有在文中出现的数据匹配不到的情况的出现。

prompt定义如下,问答的时候选的使default模式

```

"knowledge_base_chat": {
    "default":
        '<指令>根据已知信息, 简洁和专业的来判断。输出"T"或"F"表示正误, 并给出理由, 如果无法从中得到答案, 根据常识和你的思考判"T"或"F"'
        '注意不要过于抠字眼, 大概意思和关键处对了即可, 文章中没有提到的可判断为"T"'
        '不允许在答案中添加编造成分, 答案请使用中文。 </指令>\n'
        '<已知信息>{{ context }}</已知信息>\n'
        '<问题>{{ question }}</问题>\n',

    "text":
        '<指令>根据已知信息, 简洁和专业的来判断。输出"T"或"F"表示正误, 并给出理由, 如果无法从中得到答案, 请回答你认为的"T"或"F"。 </指令>\n'
        '<已知信息>{{ context }}</已知信息>\n'
        '<问题>{{ question }}</问题>\n',

    "empty": # 搜不到知识库的时候使用
        '回答你认为对则"T"反之"F":\n'
        '{{ question }}\n\n',
},

```

```

elif dialogue_mode == "知识库问答":
    chat_box.ai_say([
        f"正在查询知识库 `{selected_kb}` ...",
        Markdown("...", in_expander=True, title="知识库匹配结果",
state="complete"),
    ])
    text = ""
    for d in api.knowledge_base_chat(prompt,
                                     knowledge_base_name=selected_kb,
                                     top_k=kb_top_k,
                                     score_threshold=score_threshold,

```

```

                                history=history,
                                model=llm_model,
                                prompt_name=prompt_template_name,
                                temperature=temperature):
    if error_msg := check_error_msg(d): # check whether error
occured

        st.error(error_msg)
    elif chunk := d.get("answer"):
        text += chunk
        chat_box.update_msg(text, element_index=0)

    text = text.strip('\n').strip("\"")
    print(text)
    result.append(text[0])
    chat_box.update_msg(text, element_index=0, streaming=False)
    chat_box.update_msg("\n\n".join(d.get("docs", [])),
element_index=1, streaming=False)
```

一个问答环节的webui界面展示:



在2024年，工业互联网产业联盟全体会议期间，批准了《SPN在工业场景中的需求和应用研究报告》。In 2024, during the plenary meeting of the Industrial Internet Industry Alliance, the Research Report on the Demand and Application of SPN in Industrial Scenarios was approved.



F

根据提供的已知信息，关于《SPN在工业场景中的需求和应用研究报告》的审批是在2023年1月，而不是2024年。

✓ 知识库匹配结果

出处 [1] [useful\\_data\\_100.txt](#)

In January 2023, the Research Report on theRequirements and Application of SPN in Industrial Scenarios was approved during the plenary sessionof the Industrial Internet Industry Alliance.Transport Demand Analysis in Typical IndustryScenariosNetwork

出处 [2] [useful\\_data\\_100.txt](#)

1 SI+WIStage 2 SI+WI Stage 3 WI3GPP 5G-A（R20）时间表 3GPP6G（R21）时间表已确定时间点待确定时间点争议区间ETSI ISG MECETSI ISG5D会议通过作为6G纲领性的文件《IMT面向2030及未来发展的框架和总体目标建议书》。建议书提出了六大应用场景：沉浸式通信、超大规模连接、极高可靠低时延、人工智能与通信的融合、感知与通信的融合以

在知识库问答中，我采用了默认的prompt\_template\_name为"知识库问答"，temperature为0.7，kb\_top\_k为5。

对于大模型输出的语句，我们要去除首尾的引号空格和换行符，取出第一个字符，为'T'或'F'即为判断结果。把结果保存到一个列表里面。



## 保存结果

最后我们把所有转化成一个dataframe写入result.csv,就得到最终的结果了！

```
import pandas as pd

def lst_to_csv(lst):
    # 创建一个列表

    # 创建一个DataFrame
    df = pd.DataFrame({'id': range(1, len(lst) + 1), 'answer': lst})

    # 设置列名
    df.columns = ['id', 'answer']

    # 将DataFrame写入.csv文件
    df.to_csv('D:\\Langchain-Chatchat\\webui_pages\\dialogue\\result_b.csv',
              index=False, encoding='utf-8')

lst_to_csv(result)
```

## 总结

通过以上步骤，实现了对给定数据集的处理，包括数据清洗、数据预处理、知识库问答、结果保存等。最终得到了一个包含判断结果的.csv文件。

创新点：

- 采用了逆向检索方式通过query逆向获取文本与问题相关的信息
- 采用“罕见词”查找的检索方式，极大减小了数据集内容，并且实现了精确对位
- 使用langchain-chatchat中的知识库问答功能，并对其中一些参数（如历史记录次数，知识匹配条数）进行了实验和调整。

还有一些可以提升优化的地方可供参考：

- 对于中文和英文文档可以使用不同的chunk\_size，因为中文的一个字是两个byte，英文的一个字母就是一个byte。表达相同的内容中文的字符数会小于英文的字符数。
- 采用混合问答结果综合排序的方式，对不同方式运行产生的判断结果进行综合分析
- 构造验证集或者通过fine-tuning的方式进行微调。