1. 背景介绍

对于即将毕业大学生来说，最为关心的问题就是，“我还需要哪些技术来充实自己，以便在即将到来的就业中做好准备”。

例如，对于想要应聘java工程师岗位的同学，在大学期间已经学习了java语言和数据库相关技术，那么还需要掌握哪些技能补充自己的知识结构，才能在应聘中获得机会呢？

我已经掌握了Java相关的技术，那么哪些岗位是适合我的呢？

从技术和岗位匹配的角度还可以提问，我的c语言学得很好，那么哪些岗位是适合我去应聘的呢？

2. 实验数据

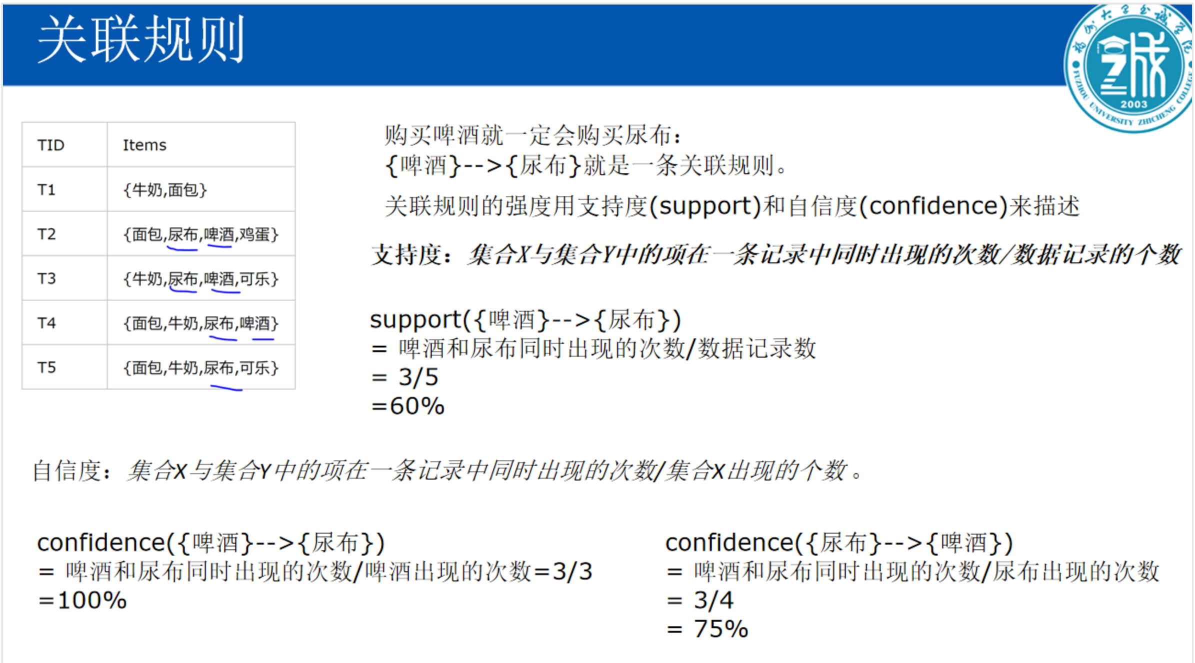
为了减少开发时间和复杂度，本次实验提供的是已经预处理好的数据集，除了部分域做了数字化外，还增加了一列技能列（skill_list)。该列是由岗位信息（positionDetail)这个非结构化文本中抽取的所需技能清单。

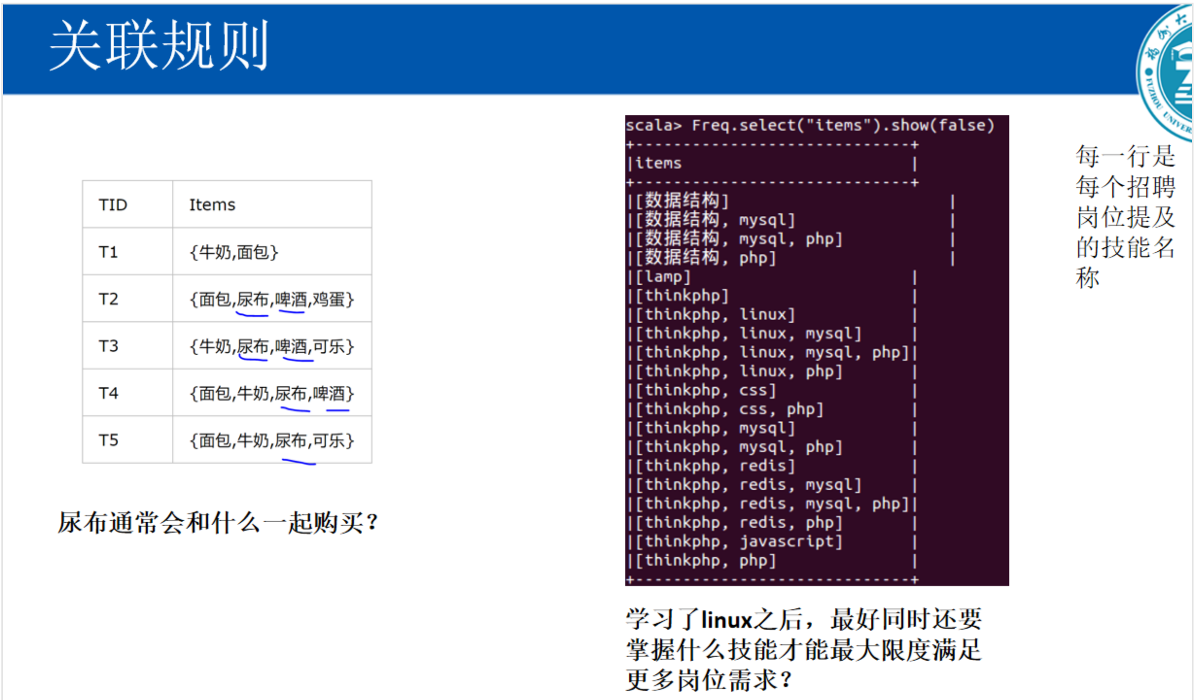
	positionName	companyFullName	companySize	createTime	city	workYear	education	positionDetail	mean_salary	workyear_num	skill_list
0	算法工程师	北京华航唯实机器人科技股份有限公司	150-500人	2022-04-29 14:16:29	北京	不限	硕士	任职资格：\n 1、机器人、自动化控制、 应用数学等相关专业；\n ...	22.5	0	机器人,c++,ros,系列,动画,仿真,解析,三维,框架
1	算法工程师	北京字节跳动网络技术有限公司	2000人以上	2022-05-07 01:47:11	北京	3-5年	本科	职位职责：\n 1、研究数据挖掘或统计学习领域的前沿技术，针对海量用户行为和内容...	30.0	4	数据挖掘,统计,前沿技术,海量,信息,特征,模型,采集,机器学习,c,c++,python,...
2	算法工程师	北京兰云科技有限公司	50-150人	2022-05-07 17:36:38	南京	3-5年	本科	岗位职责：\n 1、基于海量用户及信息系统实体行为数据，开展数据探索，挖掘潜在安...	18.0	4	海量,实体,挖掘,算法,机器学习,异常,预测,linux,python,r,深度学习,ten...

3. 实验目的

1. 掌握关联规则运用
2. 熟练运用LLM到函数调用的方法
3. 熟练运用LLM完成函数返回值到自然语言的转换

4. 背景知识





通过将招聘数据集中的招聘文本做信息抽取，获得这条招聘广告所需要的对技能的要求，基于这些技术共同出现的情况，做关联规则分析。分析技术之间的关联性。

5. 实验任务

1. 根据现有数据创建频繁集，并且在此基础上做关联规则。

2. 将关联规则作为知识库，根据用户提问，搜索关联规则，并且将结果以自然语言的形式答复给用户。我擅长java，请给我一些建议。答复应该包含和java强相关的技术以及java技术对应最多的岗位。
3. 创建技术-岗位的关联性函数
4. 实现用户使用自然语言完成三类问题的问答（输入自然语言，返回自然语言）：（1）我比较擅长xxx，请给我推荐一些关联技术（关联规则）（2）从事xxx岗位需要学什么？(从岗位查询技术)(3)我会xxx，什么岗位比较适合我？（从技术查岗位）

   注：受到实验进度的影响，同学们优先先完成一个问题的自然语言对话（从用户提问-->函数调用-->自然语言呈现）。

6. 实验步骤

6.1. 观察数据集的内容

将本次提供的数据集展开观察：

	positionName	companyFullName	companySize	createTime	city	workYear	education	positionDetail	mean_salary	workyear_num	skill_list
0	算法工程师	北京华航唯实机器人科技股份有限公司	150-500人	2022-04-29 14:16:29	北京	不限	硕士	任职资格：\n 1、机器人、自动化控制、应用数学等相关专业；\n ...	22.5	0	机器人,c++,ros,系列,动画,仿真,解析,三维,框架
1	算法工程师	北京字节跳动网络技术有限公司	2000人以上	2022-05-07 01:47:11	北京	3-5年	本科	职位职责：\n 1、研究数据挖掘或统计学习领域的前沿技术，针对海量用户行为和内容...	30.0	4	数据挖掘,统计,前沿技术,海量,信息,特征,模型,采集,机器学习,c,c++,python,...
2	算法工程师	北京兰云科技有限公司	50-150人	2022-05-07 17:36:38	南京	3-5年	本科	岗位职责：\n 1、基于海量用户及信息系统实体行为数据，开展数据探索，挖掘潜在安...	18.0	4	海量,实体,挖掘,算法,机器学习,异常,预测,linux,python,r,深度学习,ten...

注意，为了方便实验，这个数据集已经做过预处理。除了部分列转为数字之外，添加了skill_list列，将这个招聘信息中技能词汇抽取出来。（具体的实现方法：句子级别聚类-->单词级别聚类-->技术词汇清洗-->与分词后的岗位描述信息做交集）

提示：直接执行.head()会列比较多重要信息被隐藏，展示的时候选择想要看的列做展示。

6.2. 将skill_list提取出来形成二维列表

1. 观察数据：

```
# dataset自行读取，略
print(dataset['skill_list'][0])
```

   输出以逗号分割的字符串：`机器人,c++,ros,系列,动画,仿真,解析,三维,框架`

2. 将skill_list提取出来转为列表

```
lst = list(dataset['skill_list'])
#打印前5个出来看看
print(lst[:5])
```

   输出：

```
['机器人,c++,ros,系列,动画,仿真,解析,三维,框架',
'数据挖掘,统计,前沿技术,海量,信息,特征,模型,采集,机器学习,c,c++,python,java,任意,算法,
数据结构,大规模,hadoop,spark,hive',
'海量,实体,挖掘,算法,机器学习,异常,预测,linux,python,r,深度学
习,tensorflow,pytorch,sklearn,框架,数据处理',
'数据挖掘,建模,特征,模型,python,r,sql,机器学习,算法,框架,lr,scikit,learn,编程语言,数
据结构',
'数据挖掘,建模,采集,标注,java,python,任意,数据结构,算法,机器学习,深度学
习,tensorflow,caffe,mxnet,torch,框架,web']
```

3. 将lst 转为二维列表skill_list

   需要将lst中的每个成员按照逗号切割成列表，这样就得到了二维列表：

```
[['机器人', 'c++', 'ros', '系列', '动画', '仿真', '解析', '三维', '框架'],
['数据挖掘', '统计', '前沿技术', '海量', '信息', '特征', '模型', '采集', '机器学习',
'c', 'c++', 'python', 'java', '任意', '算法', '数据结构', '大规模', 'hadoop',
'spark', 'hive'],
['海量', '实体', '挖掘', '算法', '机器学习', '异常', '预测', 'linux', 'python',
'r', '深度学习', 'tensorflow', 'pytorch', 'sklearn', '框架', '数据处理'],
['数据挖掘', '建模', '特征', '模型', 'python', 'r', 'sql', '机器学习', '算法', '框
架', 'lr', 'scikit', 'learn', '编程语言', '数据结构'],
['数据挖掘', '建模', '采集', '标注', 'java', 'python', '任意', '数据结构', '算法',
'机器学习', '深度学习', 'tensorflow', 'caffe', 'mxnet', 'torch', '框架', 'web'],
....
]
```

6.3. 计算关联规则

调用Apriori算法挖掘频繁项集，推算关联规则。注意，频繁项计算耗时比较长，建议在程序启动的时候算一次（或者将计算结果保存起来，程序启动直接装载），不要每次给用户提供服务的时候临时计算。（本次实验提供了apriori.py）

可以将计算关联规则单独写入一个.py文件，不需要每次都重新计算

```
from apriori import calcu_apriori

# support表示支持度（<1的小数），自定义，根据目前数据集的情况做调整，支持度越大，计算量越小，可
以推荐的技术越少。
# 这里将计算结果保存到apriori.bin中，这样只需要计算一次关联规则即可。
big,biga = calcu_apriori(skill_list, support,savefile="apriori.bin")
```

装载并且测试关联规则：

```
# 这里的0.1可以自行调整，表示只打印满足最小的支持度的技能。这个值越大，打印的数量越少。
# linux：要查询的技术的名称，如果由用户输入，必须全部转为小写
biga_load = load_apriori("apriori.bin")
s = getAsso(biga_load, 'linux', 0.1)
print(s)
```

输出结果为：`{'python', '框架', '数据库', '编程', 'mysql'}`

6.4. 基于关联规则信息的大模型对话

实现过程如下：

1. 定义和描述回调函数
2. 输入用户的自然语言，调用LLM识别用户的语句是否匹配对应的函数，如果匹配，则将参数一起抽取出来。
3. 调用对应的函数，获得操作结果
4. 将操作结果和用户的问题一起发给LLM，请LLM将其转为自然语言

6.4.1. 封装关联规则函数

```
def get_asso_skill(dataset,skill,prompt):
    print(f"get_asso_skill: {skill}")
    biga_load = load_apriori( "apriori.bin")
    s = getAsso(biga_load, skill.lower(), 0.1)
    print(f"{skill}对应的关联技术是：{s}")
    if len(s) == 0:
        return prompt
    else:
        s = list(s)
        s = ', '.join(s)
        return f"针对问题{skill},我查询到关联性最高的岗位是{str(s)}，请将查询结果组织成人类语言描述。可以附加一些学习建议。总字数不超过200字"
```

注意：

1. 增加打印信息，以便确认后续LLM调用的时候，抽取的参数是否正确。
2. 将输入的skill参数统一转为小写
3. 如果并未查询到关联技术，则将原始提示词返回
4. 如果查询到关联技术，则将其填写到提示词模板，将新的提示词返回。

6.4.2. 定义tools

定义LLM的tools中的回调函数功能。

产生关于get_asso_skill函数的描述信息`query_asso_skill_tool`

注意： 这里并没有给出get_asso_skill的所有参数，而是只给出了需要从自然语言中抽取的参数（技能）

```
get_asso_skill_dict = {
    "name": "get_asso_skill",
    "description": "针对用户进行学习推荐。根据用户提示的技能，找出关联的技能用于学习推荐",
    "parameters": {
        "type": "object",
        "properties": {
            "skill": {
                "type": "string",
                "description": "技能名称，如java，python，c++",
            },
        },
        "required": ["skill"],
    },
```

```
}
query_asso_skill_tool = {
            "type": "function",
            "function": get_asso_skill_dict
        }
```

6.4.3. LLM识别自然语言匹配的函数

将提示词和tools放在会话里面调用LLM。（具体参考给定的例子）

```
    response = client.chat.completions.create(
        model="glm-4", # 填写需要调用的模型名称
        messages = [
            {
                "role": "user",
                "content": prompt
            },

        ],
        tools = [query_asso_skill_tool],
        tool_choice="auto",
        temperature=temperature
    )
```

每个LLM对于返回值的识别方式不一样，具体参考各自的例子。

这里需要注意，除了从大模型识别出来的参数之外，用户可以自行添加参数，以glm为例：

```
def ask_glm(client, tools, temperature, content,function_list,dataset):
    ack = chatTools(client,content,tools, temperature)

    # 调用解析和函数执行过程
    if ack.finish_reason == "tool_calls":
        tool_call = ack.message.tool_calls[0]
        function_name = tool_call.function.name
        function_args_str = tool_call.function.arguments  # Assuming it's a JSON-
encoded string
        try:
            function_args_dict = json.loads(function_args_str)  # Convert the
string to a dictionary
        except json.JSONDecodeError:
            print("Invalid JSON string encountered for function arguments.")
        else:
            print(f"function_name:{function_name}")
            if function_name not in function_list:
                print("function_name not in function_list")
            else:
                func = function_list[function_name]
                function_args_dict["dataset"] = dataset
                function_args_dict["prompt"] = content
                return func(**function_args_dict)

    return ack.message
```

这里`function_args_dict["dataset"] = dataset`这里将dataset作为参数传递进来了，类似的还有prompt

这里为了方便，做了一个function_list（学生也可以自行定义解决方法），实现函数名到函数的对应。

```
function_list = {"get_asso_skill":get_asso_skill}
```

这里可以做一个测试：

- 用户输入：`"我C++学得不错，请推荐相关的技术学习"` ==>prompt
- 将该输入和函数定义(tools)一起送给大模型（调用ask_glm）

运行结果：

```
提问：我C++学得不错，请推荐相关的技术学习
function_name:get_asso_skill
get_asso_skill: C++
C++对应的关联技术是：{'单片机', 'i2c', '操作系统', 'spi', 'freertos', '算法', 'linux', '通信', 'uart', '设备', 'arm', 'c', '编程'}
函数回调返回：针对问题C++,我查询到关联性最高的岗位是单片机，i2c，操作系统，spi，freertos，算法，linux，通信，uart，设备，arm，c，编程，请将查询结果组织成人类语言描述。可以附加一些学习建议。总字数不超过200字
```

6.4.4. 调用LLM将查询结果组织成人类语言

这个过程比较简单，只需要将上一个步骤的返回值作为提示词，调用LLM，再获取返回即可。完成的测试结果：

```
提问：我C++学得不错，请推荐相关的技术学习
function_name:get_asso_skill
get_asso_skill: C++
C++对应的关联技术是：{'单片机', 'i2c', '操作系统', 'spi', 'freertos', '算法', 'linux', '通信', 'uart', '设备', 'arm', 'c', '编程'}
函数回调返回：针对问题C++,我查询到关联性最高的岗位是单片机，i2c，操作系统，spi，freertos，算法，linux，通信，uart，设备，arm，c，编程，请将查询结果组织成人类语言描述。可以附加一些学习建议。总字数不超过200字
index=0 finish_reason='stop' message=CompletionMessage(content='查询结果显示，C++语言在以下领域关联性最高：单片机开发、I2C和SPI通信协议、操作系统（特别是FreeRTOS）、算法设计、Linux系统编程、串行通信（UART）以及基于ARM架构的设备编程。这意味着掌握这些技术对于从事C++开发工作至关重要。\n\n学习建议：重点学习C++在嵌入式系统和实时操作系统中的应用，熟悉I2C、SPI和UART等通信协议，了解Linux系统编程和FreeRTOS。同时，加强算法和数据结构的学习，提高编程效率。此外，实践是提高技能的关键，尝试参与相关项目，巩固所学知识。', role='assistant', tool_calls=None)
```

6.5. 更多回调函数

6.5.1. 查找技能对应的高频岗位

搜索position_skill，对于每一个成员，如果出现了对应的技能，则将对应的岗位名称取出放入列表。统计列表中的词频，再将词频倒序排列。

函数实现自定义。

以下实现仅供参考：

```
# dataset为当前数据集，skill是技能词汇，例如'java','linux'，count表示返回做多几个岗位。
def cacu_skill_position_wordcount(dataset, skill,count=2):
    # 将skill转为小写。
    skill = skill.lower()
    # 构建字典，方便后续排序
    postion_dict = {}
    # 遍历dataset
    for i in range(len(dataset)):
        # 将dataset中这一行的'skill_list'的内容取出来，使用逗号进行切割。将切割后的词频放入字典postion_dict做词频统计。
        #补充代码
    #将字典按照value排序，倒序排列，取前面count个成员
```

```
# 根据提供的技能，找到关联最大的岗位
skill = 'C/C++'
ret = cacu_skill_position_wordcount(dataset,skill,count = 2)
print(f"{skill} 关联最大的岗位是：{list(ret.keys())}")
```

```
C/C++ 关联最大的岗位是：['算法工程师', 'C/C++开发工程师']
```

6.5.2. 岗位到技能的关联

根据用户给定的岗位关键词，将skill_list取出，统计词频。将最高词频的若干个单词返回：

以下代码仅供参考，可以自行设计：

```
# 忽略字符串大小写做字符串对比，key如果落在str中也算匹配
def compare_str(str1, key):
    s1 = str1.lower()
    s2 = key.lower()
    if s1 == s2:
        return True
    if s2 in s1:
        return True
    return False

def cacu_postion_skill_wordcount(dataset, postionName, count=15):
    # 将postionName和数据集中'positionName'列匹配的行抽取出来，放入postion_data
    # 考虑到用户不一定会严格给出数据集相同的名称，采用自定义函数识别postionName和数据集中'positionName'列的匹配程度compare_str

    # 将postion_data的'skill_list'放入skill_list
    # 将skill_list中的每个成员都是用逗号切割，将所有切割出来的成员做词频统计

    # 将统计出来的词频字典倒序排列，取最高的count的个成员返回
```

调用测试：

```
jobstr = '嵌入式'  #'C/C++开发工程师'
word = cacu_postion_skill_wordcount(dataset,jobstr,count = 20)
print(f"{jobstr} 相关岗位最需要的技能是：{list(word.keys())}")
```

预期输出：

`嵌入式 相关岗位最需要的技能是：['c', '编程', 'c++', 'linux', '通信', 'arm', '单片机', 'spi', 'i2c', 'uart', 'mcu', '电子', '计算机', '设备', '通讯', 'usb', '操作系统', '算法', 'freertos', '外设']`

6.5.3. 将新增的两个回调函数添加到与LLM的互动中

以下三个自然语言问题就对应了本次实验的三个函数：

1. "掌握C++可以从事什么岗位？"
2. "我C++学得不错，请推荐相关的技术学习"
3. "嵌入式岗位需要什么技能？"

运行效果如下：

```
提问：掌握C++可以从事什么岗位？
function_name:cacu_skill_position_wordcount
调用cacu_skill_position_wordcount的结果：dict_keys(['C/C++开发工程师', '算法工程师'])
函数回调返回：针对问题掌握C++可以从事什么岗位？,我查询到关联性最高的岗位是dict_keys(['C/C++开发工程师', '算法工程师'])，请将查询结果组织成人类语言描述。可以附加一些学习建议。总字数不超过200字
index=0 finish_reason='stop' message=CompletionMessage(content='掌握C++可以从事的岗位主要包括C/C++开发工程师和算法工程师。这些岗位通常要求较强的编程能力和算法基础。要进一步优化技能，可以深入学习操作系统、数据结构与算法，并熟悉相关开发工具和库。同时，实践项目经验也很重要。', role='assistant', tool_calls=None)
提问：我C++学得不错，请推荐相关的技术学习
function_name:get_asso_skill
get_asso_skill: C++
C++对应的关联技术是：{'单片机', 'i2c', '操作系统', 'spi', 'freertos', '算法', 'linux', '通信', 'uart', '设备', 'arm', 'c', '编程'}
函数回调返回：针对问题C++,我查询到关联性最高的岗位是单片机, i2c, 操作系统, spi, freertos, 算法, linux, 通信, uart, 设备, arm, c, 编程，请将查询结果组织成人类语言描述。可以附加一些学习建议。总字数不超过200字
index=0 finish_reason='stop' message=CompletionMessage(content='查询结果显示，C++语言在以下领域关联性最高：单片机开发、I2C和SPI通信协议、操作系统（特别是FreeRTOS）、算法设计、Linux系统编程、串行通信（UART）以及基于ARM架构的设备编程。这意味着掌握这些技术对于从事C++开发工作至关重要。\n\n学习建议：重点学习C++在嵌入式系统和实时操作系统中的应用，熟悉I2C、SPI和UART等通信协议，了解Linux系统编程和FreeRTOS。同时，加强算法和数据结构的学习，提高编程效率。此外，实践是提高技能的关键，尝试参与相关项目，巩固所学知识。', role='assistant', tool_calls=None)
提问：嵌入式岗位需要什么技能？
function_name:cacu_postion_skill_wordcount
调用cacu_postion_skill_wordcount的结果：['c', '编程', 'c++', 'linux', '通信', 'arm', '单片机', 'spi', 'i2c', 'uart', 'mcu', '电子', '计算机', '设备', '通讯']
函数回调返回：针对问题嵌入式岗位需要什么技能？,我查询到技能重要性从高到低分别是c, 编程, c++, linux, 通信, arm, 单片机, spi, i2c, uart, mcu, 电子, 计算机, 设备, 通讯，请将查询结果组织成人类语言描述。可以附加一些学习建议。总字数不超过200字
```

```
index=0 finish_reason='stop' message=CompletionMessage(content='嵌入式岗位关键技能包括熟练掌握C和C++编程语言，深入了解Linux操作系统。此外，对通信协议如SPI、I2C、UART的掌握及ARM和单片机知识尤为重要。还需了解MCU、电子和计算机原理。建议从基础学起，先掌握C/C++编程，然后深入学习Linux和ARM架构，实践通信协议和单片机开发，逐步提升嵌入式系统设计和开发能力。',
role='assistant', tool_calls=None)
```

6.5.4. 重要的注意事项

1. 如何合理备注函数描述信息。如何使大模型能够理解用户语言实际上是对应了哪些函数的调用很重要。例如`"根据用户提示的岗位名称或者工作内容，找出对应的技能（或者技术/学习要求）"` 实验过程中可能遇到识别不到函数的情况，需要调整函数描述或者更换模型
2. 合理的参数备注信息，帮助大模型对信息抽取过程更加合理。对于岗位来说，可能原名叫做Java工程师，如果用户输入的是Java开发，那么它们是否是同一个内容呢？或者用户只输入了Java，例如搞java要学什么？注意在回调函数中，对输入参数做适当预处理，例如转为小写，替换掉一些无用的词。

7. 参考Demo

7.1. 调用百度千帆大模型

这里定义了两个函数（本次实验可以用上的，请另外自定义一个关联规则）

调用如下的百度千帆大模型中的ERNIE-3.5-8K-0205，测试函数功能:

```
import requests
import json

postion_skill_dict = {
                "name": "cacu_postion_skill_wordcount",
                "description": "根据用户提示的岗位名称或者工作内容，找出学习或者技能要求。用户仅仅给出要从事的工作或者方向也匹配这个函数",
                "parameters": {
                    "type": "object",
                    "properties": {
                        "postionName": {
                            "type": "string",
                            "description": "岗位名称，如果java工程师（输入时将工程师、开发、高级等信息去掉，只留下java关键词）",
                        },
                    },
                    "required": ["postionName"],
                },
            }

skill_postion_dict={
                "name": "cacu_skill_position_wordcount",
                "description": "根据用户给出的技能，查找匹配的岗位",
                "parameters": {
                    "type": "object",
                    "properties": {
                        "skill": {
```

```
                            "type": "string",
                            "description": "技术名，如java,C,C++,linux等",
                        },
                    }
                },
                "required": ["skill"],
            }

def main():
    # 调用input（）,让用户控制台输入字符串
    API_KEY = input("请输入API_KEY：")
    SECRET_KEY = input("请输入SECRET_KEY：")
    token = get_access_token(API_KEY, SECRET_KEY)

    url =
"https://aip.baidubce.com/rpc/2.0/ai_custom/v1/wenxinworkshop/chat/ernie-3.5-4k-
0205?access_token=" + token

    payload = json.dumps({
        "messages": [
            {
                "role": "user",
                "content": "Java开发所需要的技能是什么？"
            },
        ],
        #message中的content总长度、functions和system字段总内容不能超过8000 个字符，且不能
超过2048 tokens
        # examples:
https://cloud.baidu.com/doc/WENXINWORKSHOP/s/Llsr67q8h#%E7%AC%AC%E4%B8%80%E6%AC%A
1%E8%AF%B7%E6%B1%82
        "functions":[skill_postion_dict,
                     postion_skill_dict], # List(function)
        "temperature": 0.6,
        "top_p": 0.8,
        "penalty_score": 1,
        "disable_search": False,
        "enable_citation": False
    })
    headers = {
        'Content-Type': 'application/json'
    }

    response = requests.request("POST", url, headers=headers, data=payload)
    # 将JSON字符串解析为Python字典
    response_data = json.loads(response.text)
    # 检查是否触发了函数调用
    if 'function_call' in response_data:
        function_call = response_data['function_call']
        function_name = function_call['name']
        arguments = json.loads(function_call['arguments'])  # 将arguments字符串解析
为Python字典
        print(f"触发了函数调用: {function_name}")
        print(f"参数: {arguments}")


    print(response.text)
```

```
def get_access_token(API_KEY, SECRET_KEY):
    """
    使用 AK，SK 生成鉴权签名（Access Token）
    :return: access_token，或是None(如果错误)
    """
    url = "https://aip.baidubce.com/oauth/2.0/token"
    params = {"grant_type": "client_credentials", "client_id": API_KEY,
"client_secret": SECRET_KEY}
    return str(requests.post(url, params=params).json().get("access_token"))


if __name__ == '__main__':
    main()
```

运行过程中，会提示输入APIkey和SecretKey。如果嫌麻烦可以将这两个key写死在代码中，提交作业时删除即可。
以下是运行结果：

```
触发了函数调用: cacu_postion_skill_wordcount
参数: {'postionName': 'Java开发'}
{"id":"as-
dea88xadsw","object":"chat.completion","created":1714916425,"result":"","is_trunc
ated":false,"need_clear_history":false,"function_call":
{"name":"cacu_postion_skill_wordcount","thoughts":"用户想要了解Java开发所需要的技能，
这需要从岗位需求的角度来看，可以通过[cacu_postion_skill_wordcount]工具来获取信
息。","arguments":"{\"postionName\":\"Java开发
\"}"},"finish_reason":"function_call","usage":
{"prompt_tokens":365,"completion_tokens":49,"total_tokens":414}}
```