设计方案

目录

1. Text2SQL

1.1 题目任务:

将自然语言的查询语句转化成数据库能够执行的 sql 语言。

1.2 问题分析:

Text2SQL 的发展轨迹已经从基于规则的、特定领域的解决方案发展到基于深度学习的模型，特别是序列到序列模型。最近基于大型语言模型（LLM）的解决方案将性能进一步推到了一个新的高度。现有文献中已经有了许多比较好的结果，如 C3、DIN-SQL 以及 PET-SQL 等。

由于本赛题被如下:

1、 每次的 prompt 必须被限制在 2048 个 token。

2、 至多与模型进行 5 次交互。

3、 最后一次的模型输出必须是可执行的 SQL 命令。

4、 不允许使用除 JOSN 外其他库。

等规则所限制。直接应用上述所提到的现有方法是困难的。

1.3 解决方案:

我们提出的解决方案如下图所示。

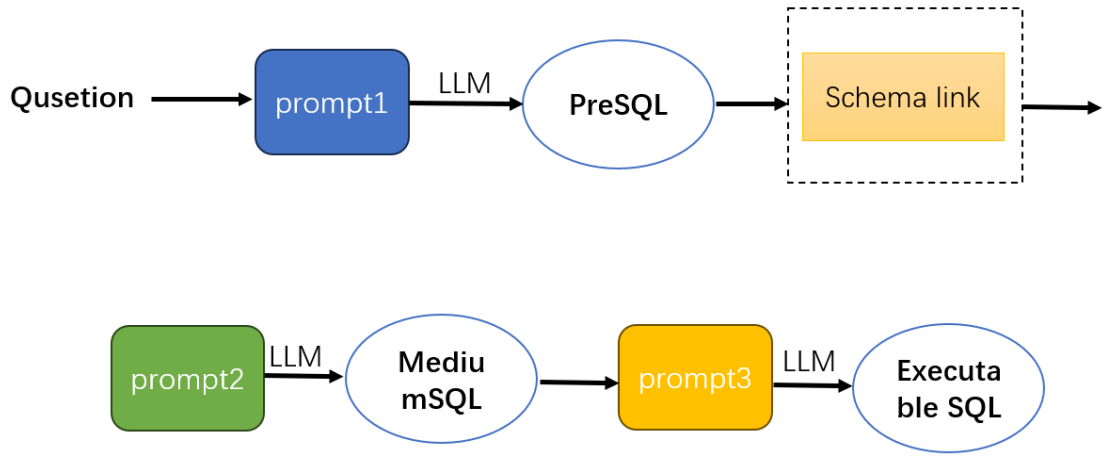

图 1 text2sql 方案

可以看到本方案通过与模型进行了三次交互，最终得到了相应问题的 SQL 查询语句，接下来将从这三个 prompt 的设计具体说明。

1.3.1 PreSQL

在 Text2SQL 中，一个重要的概念是 schema link，它可以指示和缩小 LLM 应该搜索哪些表和哪些列的答案，减小模型的幻觉，是 Text2SQL 的关键子任务。通俗的讲，shcema link 是提取出问题所需要的表，列已经具体的条件值，可以减少传递给模型的无用信息。

在 DIN-SQL 中采用的方法是通过 few shot CoT 的方法直接让模型生成 schema-link。这里如果我们直接采用 DIN-SQL 中的方法的困难主要在于:

1、本赛题对于 token 的限制，这导致给的示例将被限制在 5 个以内，少量的示例直接导致模型直接根据问题以及数据库详细信息生成 schema link 的效果很差，直接影响后续的正确率。

2、实际上相比于生成 schema link，模型更擅长生成 SQL 命令。

基于以上分析，本方案的 prompt1 设计的目的是为了让模型能根据问题以及数据库信息生成一个初步的 SQL 命令，我们在这一步 prompt 生成直接使用 zero shot。此外 prompt 的样式对于模型的表现影响十分显著，在这里参考了 PET-SQL 中的 prompt 模板，这对于模型回复的影响

从中提取出 schema link。Prompt 设计如下:

```
        user_prompt_1 = f'''### Answer the question by SQLite SQL query only and with no explanati
    You must minimize SQL execution time while ensuring correctness.
    ### Sqlite SQL tables, with their properties:
    #
    {fields_1}
    #
    ### Foreign key information of SQLite tables, used for table joins:
    #
    {foreign_keys_1}
    #
    ### Question:{user_question}
    ### SQL:
    '''         #第一步生成一个初始的schema link
```

详细的 prompt1 可以在 38 到 129 行中找到。注意我们在这里以及在本方案的后续均省略了 system 的 prompt1 设计，它在我们的源码中均在 user prompt 的上下文中可以直接找到。

1.3.2 Medium SQL

根据上一步中我们已经初步生成的 SQL 命令，我们要得到我们想要的 schema link。具体来说，schema link 的得到我们是根据 PreSQL 直接筛选了原数据库和外键，将 PreSQL 中未涉及到的表和外键直接删掉。这能有效的减轻模型幻觉。prompt2 的设计实际上就是在 prompt1 上面将数据库和外键替换为删改后的表和外键。本步的具体过程请参考源码 130 到 235 行，至此我们得到了 Medium SQL。

1.3.3 Executable SQL

由于题目要求最后生成的 SQL 命令是要由模型直接给出的，所以我们并不能像之前一样提取出中间 SQL 命令。我们这里利用 few shot 的技巧让模型提取出 Final SQL 中的可执行的 SQL 命令。Prompt3 请查看源码中的 241 到 301 行。

2. 数据库通识类选择题

2.1 题目任务

形式为单项选择题。其中，单项选择题有 4 个选项，仅一个选项是正确的。

2.2 解决方案

对于本题，我们主要使用 few shot CoT 的方法。但由于 token 的限制，我们不可能对所有题型都给出示例，为了缓解 token 的压力并且让给的示例更具有针对性，我们将选择题的题型细分为两类：

1. SQL 通识类（与 SQL 概念和操作符相关的基础知识问题）

2. 选择正确 SQL 语句类（要求选择正确的 SQL 语句来执行某个特定操作）

之后针对不同的题型在第二步交互中设计不同的 prompt 以提升模型表现。第三步从第二步的分析中提取出答案选项。第四步则是规范化输出为(A/B/C/D)。

基于上述考虑，我们的 prompt 设计方案如下图所示：

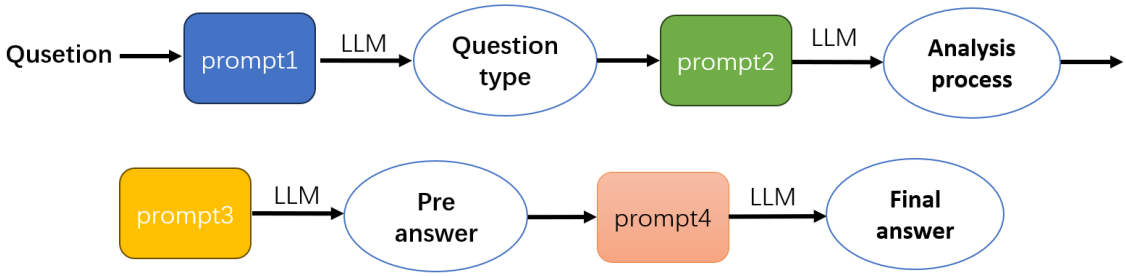

图 2 选择题方案

可以看到是通过四次与模型的交互最终得到答案，接下来就 4 次交互说明我们的 prompt 设计。

2.2.1 Question type

首先，第一步我们希望得到问题的类型，1、SQL 通识类，2、选择正确 SQL 语句类。这里我们会给出一些 few shot 示例引导模型，比如:

```
示例1:
### 问题
在SQL中,与“NOT IN”等价的操作符是?

### 选项
A. <> ALL
B. <> SOME
C. = SOME
D. = ALL

### 分类
SQL通识类
### 分类依据
这个问题涉及SQL操作符的基础知识，要求理解各个操作符的意义和用法。
```

可以看到我们不仅要求模型给出分类，而且要求其给出分类的依据，这可以有效的防止模型生成错误的分类。具体的 prompt1 内容请查阅源码的 304 到 430 行。

2.2.2 Analysis process

在得到问题的分类后，在第二次与模型交互时，我们通过应用 few shot CoT 的方法引导模型逐步分析选择题，具体来说，我们引导模型按照给的格式全面分析了四个选项，最终给出最符合题意的选项。对于问题类型的不同，实际上分析过程是相同的，只是针对性的给了不同的示例。这里给出一个 SQL 通识类示例如下:

```
问题:在SQL中，语句ALTER DATABASE属于以下哪类功能?
选项:
A. 数据查询
B. 数据操纵
C. 数据定义
D. 数据控制

回答:
选项A：数据查询是指从数据库中检索数据的操作，通常使用SELECT语句。ALTER DATABASE不是用于数据查询的。因此，这个选项不正确。

选项B：数据操纵是指插入、更新、删除和合并数据的操作，通常使用INSERT、UPDATE、DELETE等语句。ALTER DATABASE不是用于数据操纵的。因此，这个选项不正确。

选项C：数据定义是指定义或修改数据库结构的操作，通常使用CREATE、ALTER、DROP等语句。ALTER DATABASE用于修改数据库的结构或属性，因此属于数据定义操作。这是正确答案。

选项D：数据控制是指控制对数据库访问权限的操作，通常使用GRANT、REVOKE等语句。ALTER DATABASE不是用于数据控制的。因此，这个选项不正确。

最符合题意的选项为: C
```

具体 prompt2 内容请查阅源码的 439 到 752 行。

2.2.3 Pre answer

由于最终模型的输出需要为纯选项字母，所以我们在第三次交互时，是一次“答案提取”，我们通过 prompt 设计让模型根据上一步的分析，将答案选项提取出来，具体来说使用了 few shot 的方法。本步 prompt3 内容请查阅源码 756 到 776

行。

2.2.4 Final answer

为了消除第三步生成的 Pre answer 中可能含有“.”等标点符号，生成符合规范的结果(仅含 A/B/C/D)。我们添加了第四步的格式化。prompt4 内容请查看源码的 779 到 807 行。

3. 数据库通识类判断题

3.1 题目任务

形式为判断题。其中，判断题仅需要判断题目的描述是否正确。给出 True 或者 False。

3.2 解决方案

在这一题中我们通过 3 次与模型的交互得到最终结果。具体来说，第一次交互应用了 few shot CoT 方法用于分析得到判断结果，第二次根据上一次模型输出的分析结果得到判断结果，第三步规范答案。基于上述考虑，我们的 prompt 设计方案如下图所示：

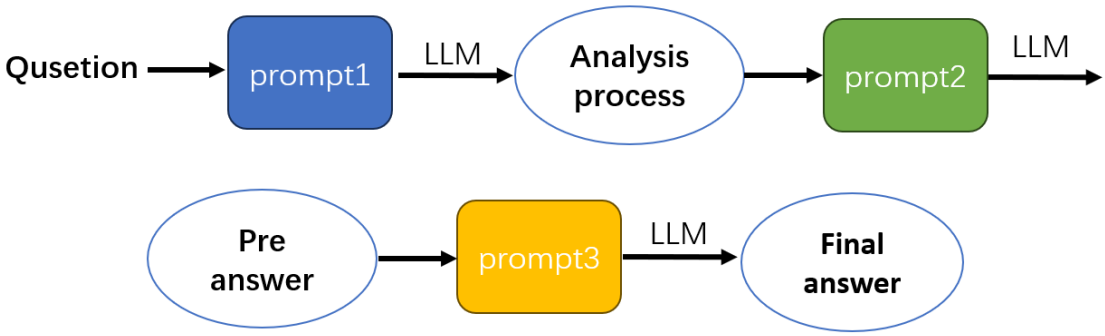

图 3 判断题方案

3.2.1 Analysis process

在判断题中，我们的核心步骤是第一步，我们通过使用 few shot CoT 的方法引导模型逐步分析判断题，并最终给出判断结果。具体来说我们引导模型通过 4 步分析问题：

1、 确认问题陈述
2、 提供与题目相关的 SQL 知识
3、 逐步分析题目是否正确
4、 给出最终结论（True 或 False）

这样的分为四步的引导模型进行分析可以有效减轻模型误判的可能。一个在 prompt 示例如下所示：

```
判断题："使用 LIKE 运算符时，可以使用 % 来匹配任意字符的任意数量。"

分析步骤：
1. **确认题目陈述：**
   - 使用 LIKE 运算符时，可以使用 % 来匹配任意字符的任意数量。

2. 提供与题目相关的 SQL 知识：
   - SQL 是结构化查询语言，用于管理和操作关系数据库。
   - LIKE 运算符用于在 SQL 中进行模式匹配。
   - % 是一个通配符，用于匹配任意数量的字符，包括零个字符。

3. 逐步分析题目是否正确：
   - 在 SQL 中，LIKE 运算符确实可以用于进行模式匹配。
   - % 作为通配符，在 LIKE 运算符中使用时，可以匹配任意数量的字符，包括零个字符。
   - 因此，陈述"使用 LIKE 运算符时，可以使用 % 来匹配任意字符的任意数量"是正确的。

4. 给出最终结论（True 或 False）：
   - 基于以上分析，这个陈述是正确的。因此，答案是：True
```

详细的 prompt1 内容请查阅源码的 810 到 978 行。

3.2.2 Pre answer

由于最终模型的输出需要为 True 或者 False，所以我们在第二次交互时，与选择题相同是一次“答案提取”，我们通过 prompt 设计让模型根据上一步的分析，将答案选项提取出来，具体来说也使用了 few shot 的方法。本步 prompt2 内容请查阅源码 980 到 991 行。

3.2.3 Final answer

为了消除第二步生成的 Pre answer 中可能含有“.”等标点符号，生成符合规范的结果(True/False)。我们添加了第三步的格式化。prompt3 内容请查看源码的 995 到 1004 行。