Corrector 实现解析

一、背景介绍

介绍 Corrector 以前，先简单介绍一下 SuperSonic 主要流程以及 LLMS2SQLParser 解析说明；

1.1、SuperSonic 主要流程

a. 原始文本先经过 Schema Mapper 生成 SchemaMapInfo；
b. 经过 Schema Parser 生成 SemanticQuery 和 SemanticParseInfo（包含 S2SQL）；其中规则/LLM 解析均生成 S2SQL；
c. S2SQL 经过 SemanticCorrector 对 S2SQL 进行修正，生成修正 S2SQL；
d. 修正后的 S2SQL 再经过 Semantic 层转换为执行 SQL，查询物理数据模型获取结果数据；

1.2、LLMS2SQLParser 解析参数说明：

输入参数：主要包含查询文本、识别的 value 取值 linking、识别的 modelName、以及相关的中文字段名；
输出参数：主要是大模型解析后的 S2SQL，如示例：

```
SELECT COUNT(*)  FROM 超音数 WHERE 访问者 = 'alice' AND 访问日期 >= '2023-11-04'
```

1.3、Corrector 背景

经过规则/LLM 解析生成的 S2SQL（尤其是 LLM 解析），常常出现不符合语义、信息展示不全、数据日期缺失等问题；Semantic Corrector 主要功能是 S2SQL 进行修正，使得查询结果更符合用户要求；

二、Corrector 实现解析

SuperSonic 中的 Corrector 主要有以下几类，SchemaCorrector、SelectCorrector、WhereCorrector、GroupByCorrector、HavingCorrector；采用 Spring SPI 机制按顺序加载 Corrector 实现类，可在 META-INF/spring.factories 中动态修改；除第一个 SchemaCorrector，其他主要是按照 sql 的区域进行修正；SchemaCorrector 主要是对 Schema 的数据进行修正，如字段名、value 值识别错误等；下面我们来详细讲解各个 Corrector 的实现；

2.1 SchemaCorrector

由于大模型生成的 S2SQL，在字段名、维度值、以及聚合函数等上生产错误；该 Corrector 主要作用是修正在 Schema 相关侧的 S2SQL 的错误；

a. 字段名修正：

如上面例子，S2SQL 中，访问者 = 'alice' 和访问日期 >= '2023-11-04' ；传给大模型的 filedNameList 中，不存在字段名"访问者"、"访问日期"；因此，需要将 S2SQL 中的字段"访问者"修正为"用户名"，字段"访问日期"修正为"数据日期"；

b. 维度值修正：

S2QL 的维度值，如访问者 = 'alice' 中的 alice，模型有时候会生成错误的维度值如"Alice"等；维度值错误，将导致无法查询到准确的数据；因此，需要将 S2SQL 中的维度值进行修正；

c. 修正策略：

目前的修正策略主要是利用 hanlp 来进行维度值修正，通过文本相似度来修正字段名；目前正在实现通过语义来修正字段名和维度值；

2.2 SelectCorrector

该 corrector 主要作用是将一些 S2SQL Where 条件中的字段一起展示出来，提升用户体验；如"按部门统计，访问次数大于 10 次部门有哪些"，大模型生成的 S2SQL：

```
SELECT 部门 FROM 超音数 GROUP BY 部门 HAVING SUM(访问次数) > 10
```

该查询只会将字段"部门"查出，通常用户需要知道超过 10 次的"部门"以及对应"访问次数"具体是多少；主要实现原理是通过 jsqlparser 框架将 group by、order by 涉及的字段，加入到 select 中；具体实现细节代码：com.tencent.supersonic.chat.corrector.SelectCorrector；

2.3 WhereCorrector

该 corrector 主要作用是对 S2SQL 中的 Where 部分进行修正和优化，主要有 3 个优化：

1、添加"数据日期"字段过滤

如前面的输入"按部门统计，访问次数大于 10 次部门有哪些"，返回的 S2SQL 并没有加上"数据日期"过滤，默认会全表扫描；数据日期添加的逻辑如下：

a. 判断此次查询是查询模式：指标模式 or 标签模式；具体代码参考：com.tencent.supersonic.chat.parser.QueryTypeParser；

b. 如果是指标模式，则获取问答设置中的指标模式时间范围；如果没有配置或配置的是负数，则不进行"数据日期"添加；

c. 如果是标签模式，则获取问答设置中的标签模式时间范围，其他同 b；

2、解析相对时间

对于相对时间识别，大模型识别比较复杂；SuperSonic 实现逻辑是：将相对时间 currentDate 加入到 prompt 中，并采用 datediff 来表达相对时间：

因此该 Corrector 的另外一个功能是，将约定的相对时间转换为数据库能执行的时间；

1、最近 3 天：datediff('day', 数据日期, '2022-11-06') <= 3　　转换为：数据日期 <="2022-11-06" and 数据日期 >="2022-11-04"2、最近 12 个月：datediff('month', 数据日期, '2022-11-06') <= 123、最近 3 年：datediff('year', 数据日期, '2022-11-06') <= 3

3、添加 QueryFilter 限定条件

在 QueryReq 中有一个 queryFilters 参数，主要是在某些场景，需要将已知的限定条件带到后端；如：已经明确识别到了某些条件，需要将这些限定条件加入到查询中；因此，该功能主要利用 jsqlparser 将限定条件加入到 S2SQL Where 条件中；

2.4　GroupByCorrector

该 corrector 主要作用是对大模型生成的 S2SQL 的 group by 部分进行修正；以文本-"访问次数最高的部门"举例说明，大模型生成的 S2SQL: SELECT 部门 FROM 超音数 WHERE 数据日期 = '2023-11-27' ORDER BY 访问次数 DESC LIMIT 1，不符合用户查询语言，没有对部门进行 group by；

1、大模型生成的 S2SQLSELECT 部门 FROM 超音数 WHERE 数据日期 = '2023-11-27' ORDER BY 访问次数 DESC LIMIT 12、经过 SchemaCorrectorSELECT 部门 FROM 超音数 WHERE 数据日期 = '2023-11-27' ORDER BY 访问次数 DESC LIMIT 13、经过 SelectCorrectorSELECT 部门, 数据日期, 访问次数 FROM 超音数 WHERE 数据日期 = '2023-11-27' ORDER BY 访问次数 DESC LIMIT 14、经过 WhereCorrectorSELECT 部门, 数据日期, 访问次数 FROM 超音数 WHERE 数据日期 = '2023-11-27' ORDER BY 访问次数 DESC LIMIT 15、经过 GroupByCorrectorSELECT 部门, 数据日期, SUM(访问次数) FROM 超音数用户部门 WHERE 数据日期 = '2023-11-27' GROUP BY 部门, 数据日期 ORDER BY SUM(访问次数) DESC LIMIT 1

可以看到经过 GroupByCorrector 后，我们增加了对"部门"和"数据日期"的 group by 操作，并且是对指标进行 SUM 后排序；最终结果是符合预期；

2.5 HavingCorrector

该 corrector 主要作用是对大模型生成的 S2SQL 的 Having 部分进行修正；以文本-"访问次数大于 10 次的部门有哪些"举例说明，大模型生成的 S2SQL:SELECT 部门 FROM 超音数 WHERE 访问次

数 > 10；没有按照 group by 进行聚合，并且指标过滤也需要通过 having 才可行；

```
1、大模型生成的 S2SQLSELECT 部门 FROM 超音数 WHERE 访问次数 > 102、经过 SchemaCorrectorSELECT 部门 FROM 超音数 WHERE 访问次数 > 103、经过 SelectCorrectorSELECT 部门 FROM 超音数 WHERE 访问次数 > 104、经过 WhereCorrectorSELECT 部门 FROM 超音数 WHERE (访问次数 > 10) AND 数据日期 = '2023-11-20'5、经过 GroupByCorrectorSELECT 部门 FROM 超音数用户部门 WHERE (SUM(访问次数) > 10) AND 数据日期 = '2023-11-20' GROUP BY 部门 6、经过 HavingCorrectorSELECT 部门, SUM(访问次数) FROM 超音数用户部门 WHERE 数据日期 = '2023-11-20' GROUP BY 部门 HAVING SUM(访问次数) > 10
```

可以看到经过 HavingCorrector 后，我们在 HAVING 中增加了 "SUM(访问次数) > 10 "过滤，满足了用户查询语义；

三、总结

SuperSonic 主要是通过抽象出 Corrector 接口，提供插件化灵活改造；对大模型、规则生成的 S2SQL 统一进行修正（暂关闭规则的修正），使得查询更符合用户的语义；