数据库的自然语言接口关键技术研究

胡玮文

华南理工大学

2020/6/2

目录

目录

数据库的自然语言接口

日常生活中，人们每天都在和无数数据库打交道

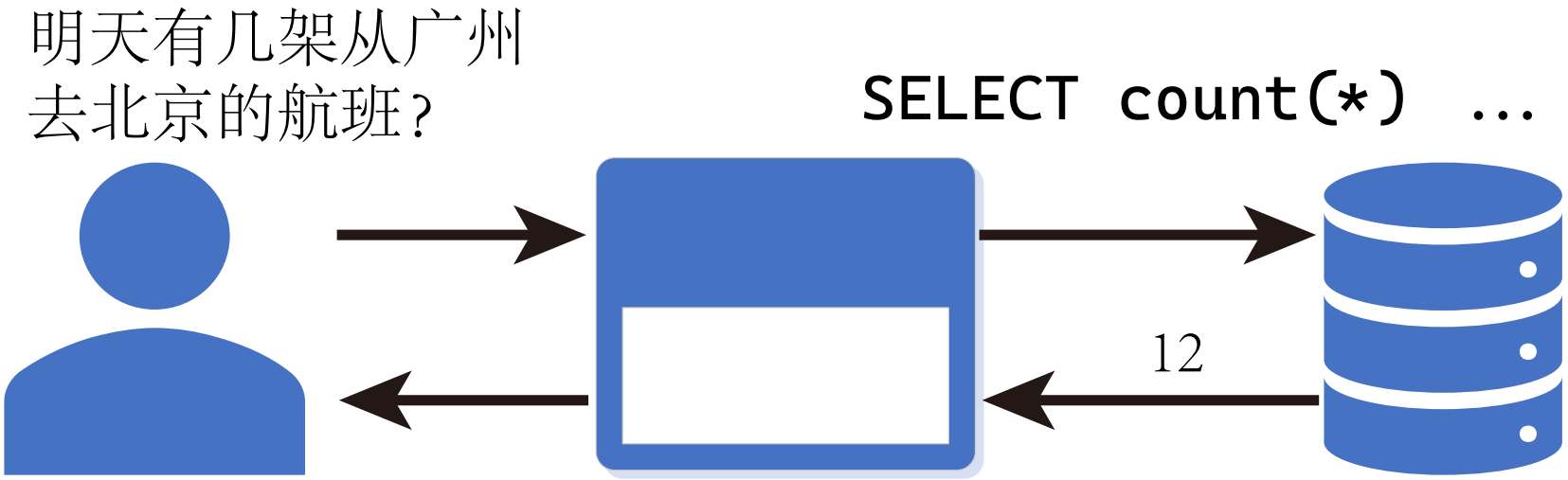

图: 当前与数据库交互的方式

数据库的自然语言接口

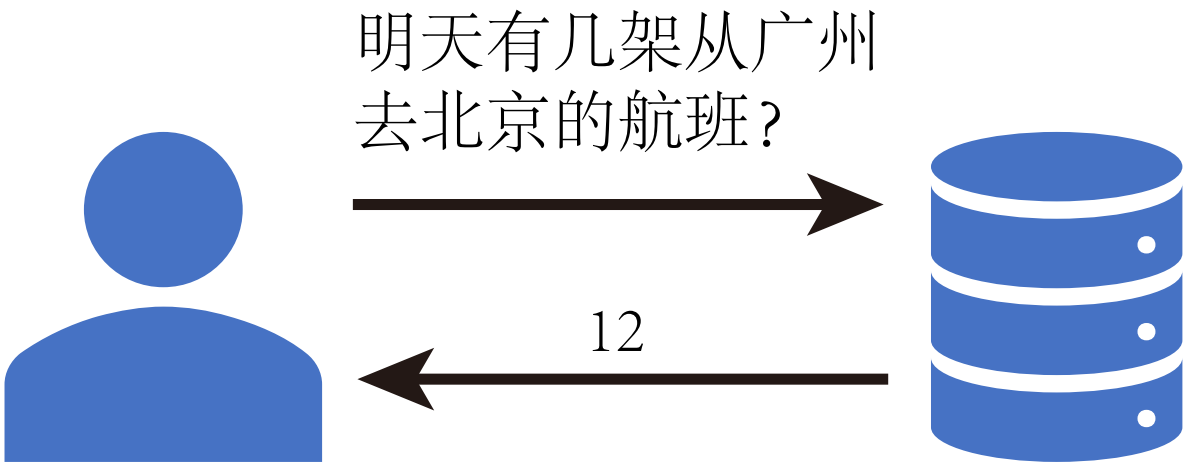

图: 通过自然语言接口与数据库直接交互

意义

广大用户首次获得了直接面对大数据的能力

自然语言到 SQL 转换任务

实现数据库的自然语言接口的方式之一：将自然语言转换为 SQL

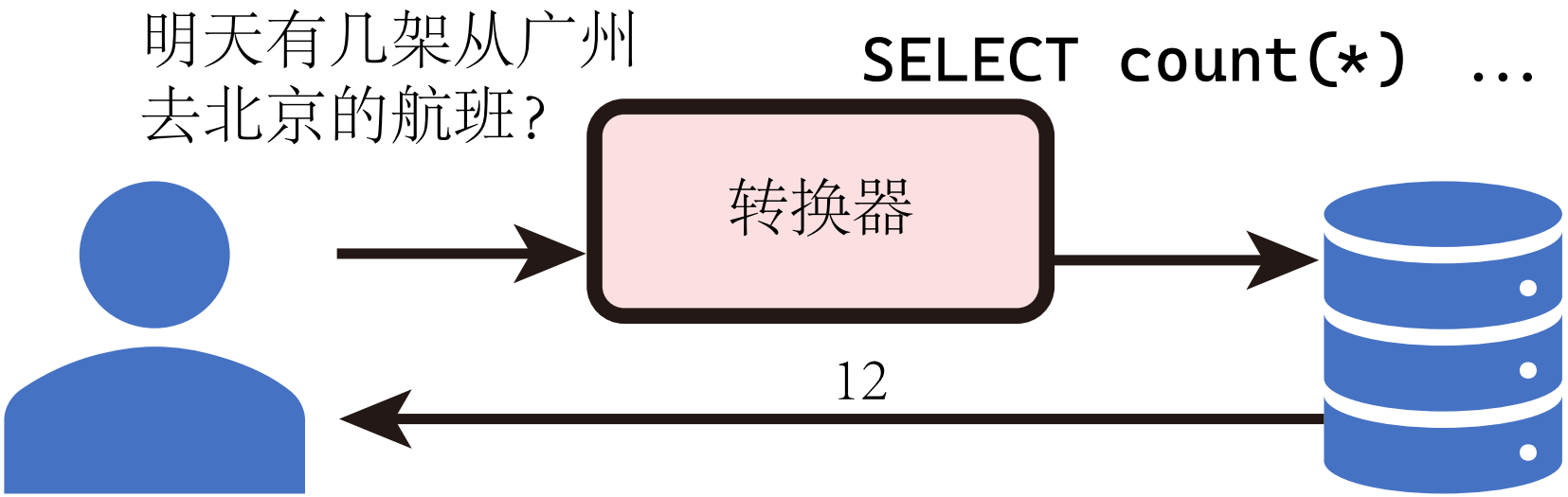

图: 通过自然语言到 SQL 转换与数据库交互

SParC 数据集

SParC 数据集是跨领域的，上下文相关的自然语言到 SQL 转换任务数据集

What are all the airlines?

```
SELECT * FROM AIRLINES
```

Of these, which is Jetblue Airways?

```
SELECT * FROM AIRLINES WHERE
Airline = "JetBlue Airways"
```

What is the country corresponding it?

```
SELECT Country FROM AIRLINES WHERE
Airline = "JetBlue Airways"
```

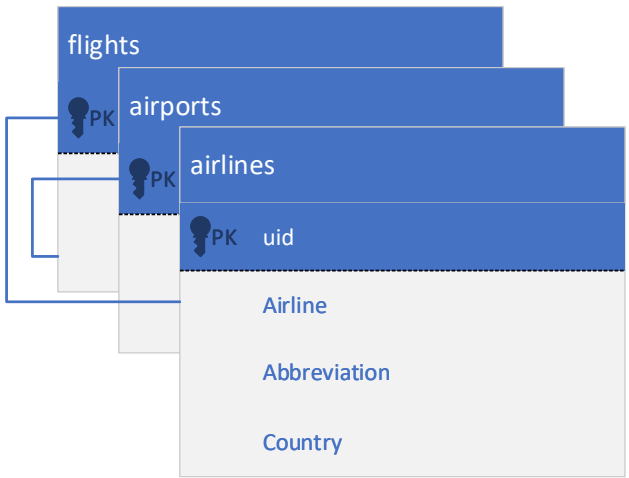

图: 数据库架构

目录

整体架构

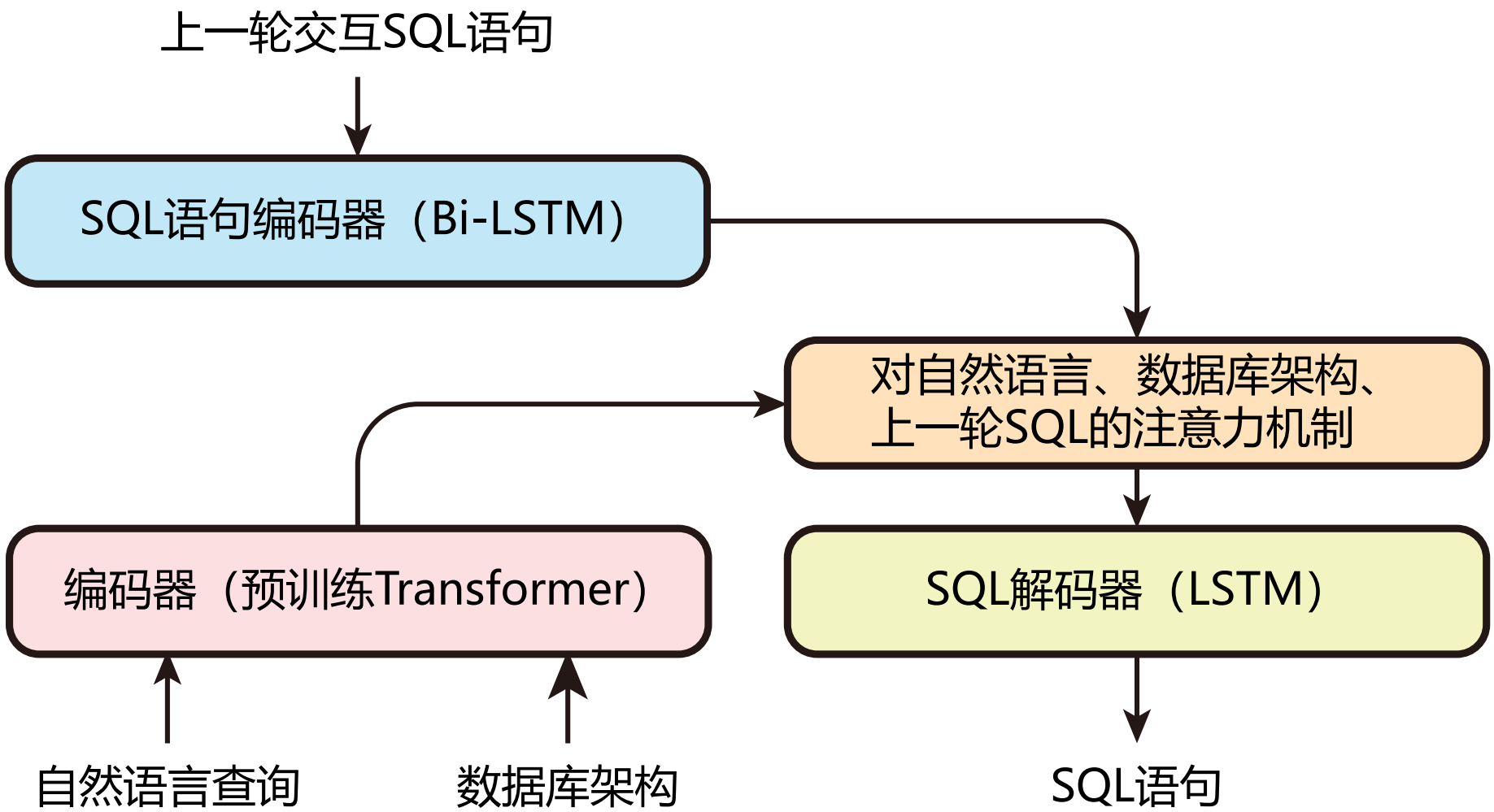

图: 模型整体架构

SQL 预处理和后处理

- SQL 中包含冗余的，抽象层次低的结构，将显著影响模型预测的准确率
- EditSQL 的方案为去除 SQL 中最为冗余的 FROM 子句

例

Who are all the party hosts?···

Show the themes of parties they host along with their name.

```
SELECT T3.Party_Theme, T2.Name FROM party_host AS T1
    JOIN host AS T2 ON T1.Host_ID = T2.Host_ID
    JOIN party AS T3 ON T1.Party_ID = T3.Party_ID
```

SQL 预处理和后处理

本文方案

预处理过程

1. 解析列引用，将引用表达式替换为“表名. 列名”的规范形式
2. 去除所有 JOIN 子句中的 ON 部分
3. 去除 FROM 子句中用于多对多关联的 JOIN 子句
4. 在 FROM 子句中，去除所有在 SQL 的其他部分引用过的表。若所有表都被去除，则去除整个 FROM 子句。

后处理过程

1. 将所有在 SQL 中引用过的表添加到 FROM 子句中。
2. 根据外键关系，将数据库中的所有表构造成无向图，并使用 Kruskal 算法求解包含当前 FROM 子句中的表的最小生成树，根据生成树的边和节点来重建 JOIN 子句中的 ON 部分。
3. 有多张表时，恢复别名

例

```
SELECT T3.Party_Theme, T2.Name FROM party_host AS T1
     JOIN host AS T2 ON T1.Host_ID = T2.Host_ID
     JOIN party AS T3 ON T1.Party_ID = T3.Party_ID
```

转换为

```
SELECT party.Party_Theme, host.Name
```

- 解析别名 T2，T3 为真正的表名
- party_host 是多对多关联表，host 和 party 表均在 SELECT 子句中被引用过，所以整个 FROM 子句全部被去除。

上下文编码机制

EditSQL 使用会话级别 RNN 编码上下文信息

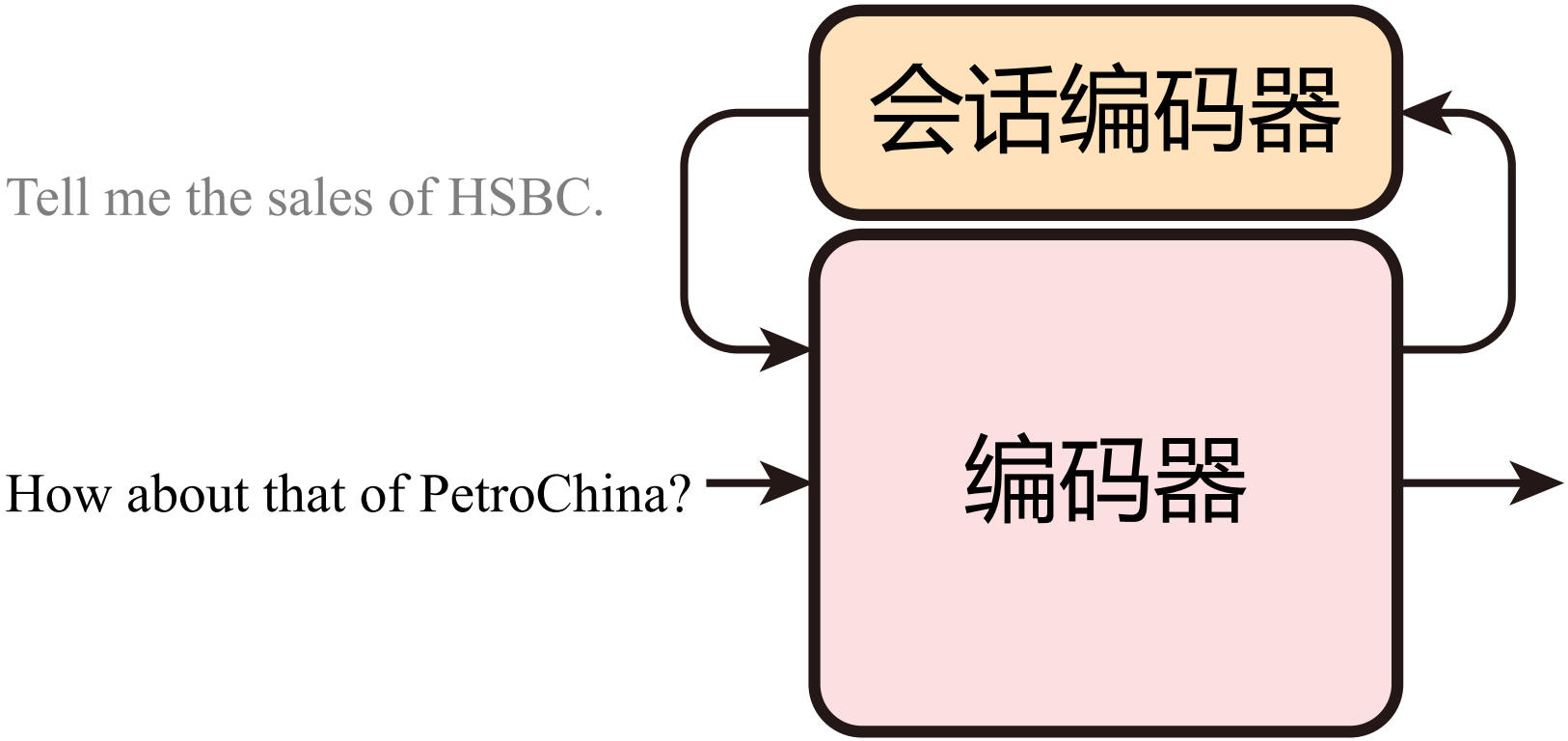

图: EditSQL 编码上下文方案

上下文编码机制

BERT 等模型经过大规模无标注文本数据预训练，能更好地理解自然语言的上下文信息，如指代、省略等现象。

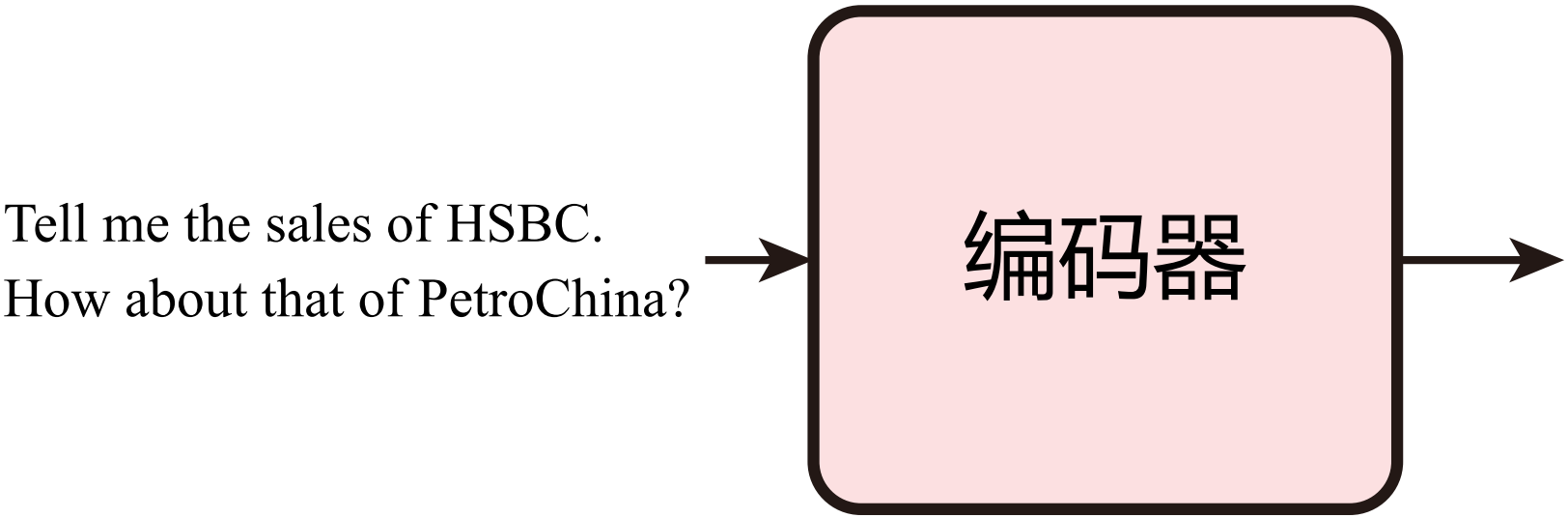

图: 本文上下文编码机制

关系编码机制

- 数据库架构中包含有主键，外键等信息。这些信息被以往的很多方法忽略
- 本文将数据库架构的编码结合进预训练注意力机制中

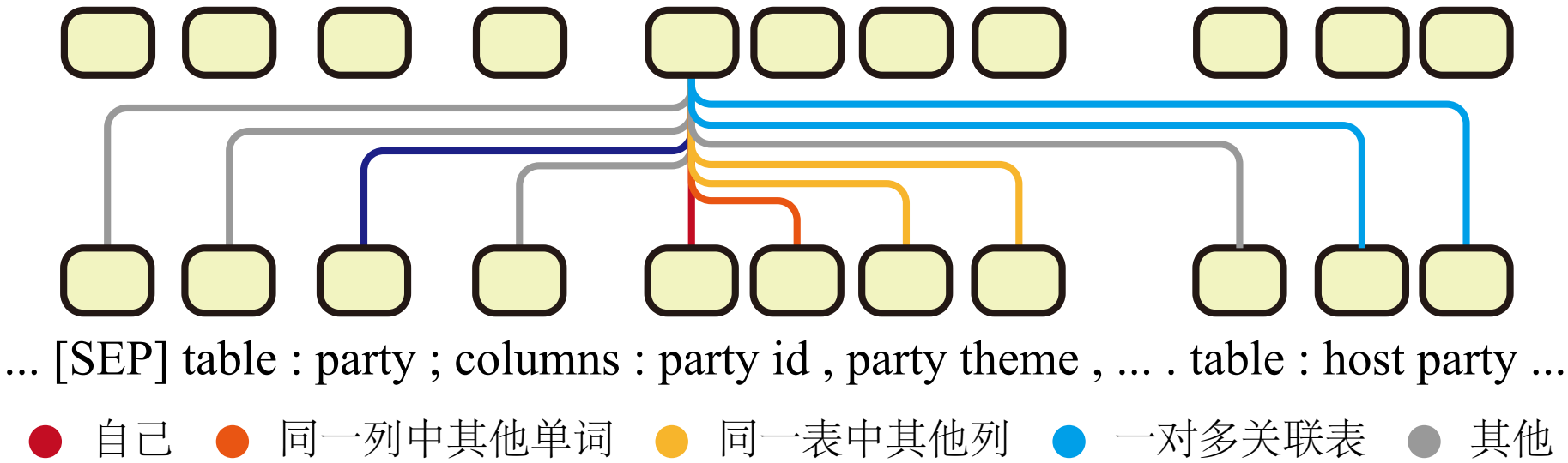

图: 关系编码机制示意[1]

[1]为了展示简洁，标点符号对应的表示已省略，它们全部使用“其他”关系

目录

总体结果

	问题准确率	交互准确率
SyntaxSQL-con	18.5	4.3
CD-Seq2Seq	21.9	8.1
EditSQL with BERT	47.2	29.5
本文模型 with BERT	54.3	34.6
本文模型 with XLNet	**58.5**	**39.6**
使用标注的历史 SQL 查询		
EditSQL with BERT	53.4	29.2
本文模型 with BERT	60.7	34.6
本文模型 with XLNet	**64.3**	**39.3**

表: SParC 实验总体结果

消融实验

SQL 预处理和后处理

- “无 FROM”为 EditSQL 所用方案
- “FROM 未引用表”为本文最终采用方案

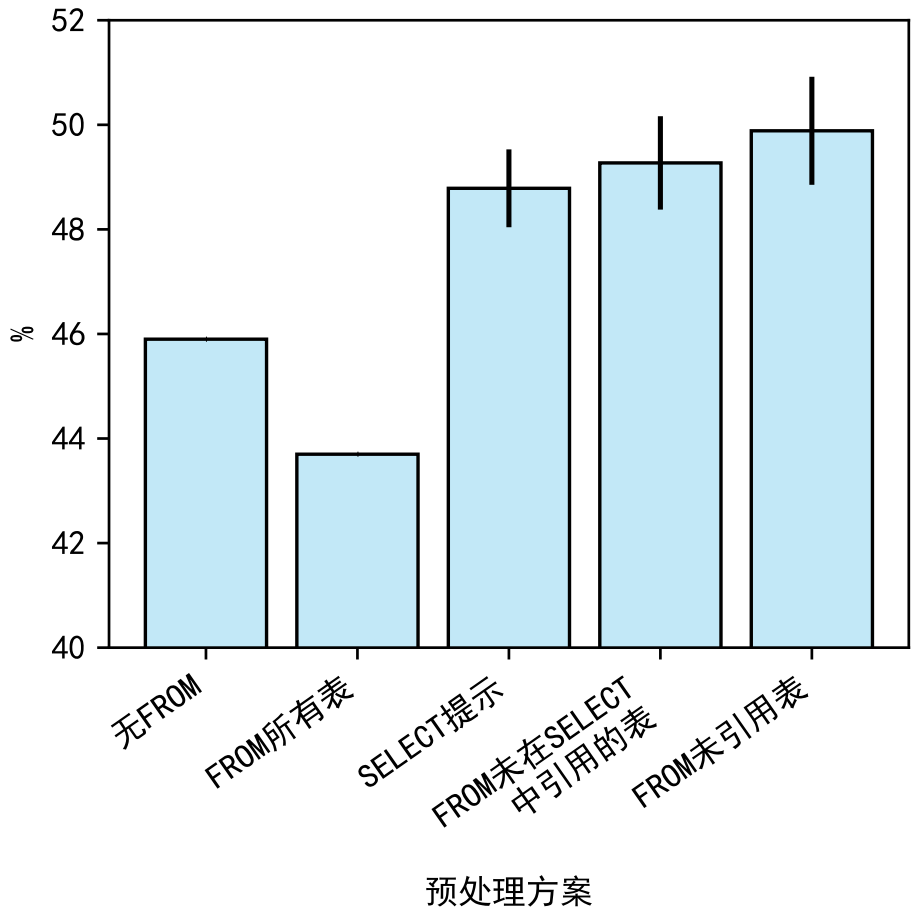

图: 各种预处理方案对比

消融实验

上下文编码 & 关系编码机制

	问题准确率	交互准确率
本文模型	58.5	39.6
- 上下文编码	54.2	33.9
- 关系编码	57.1	38.9

表: 消融实验结果

实验对比结果

- 上下文编码机制虽然简单，但效果显著
- 关系编码机制带来较小提升

错误分析

- 大多数错误出现在单个子查询内
- 较高层次的查询骨架错误和较低层次的列选择错误都较多
- 语法错误几乎没有

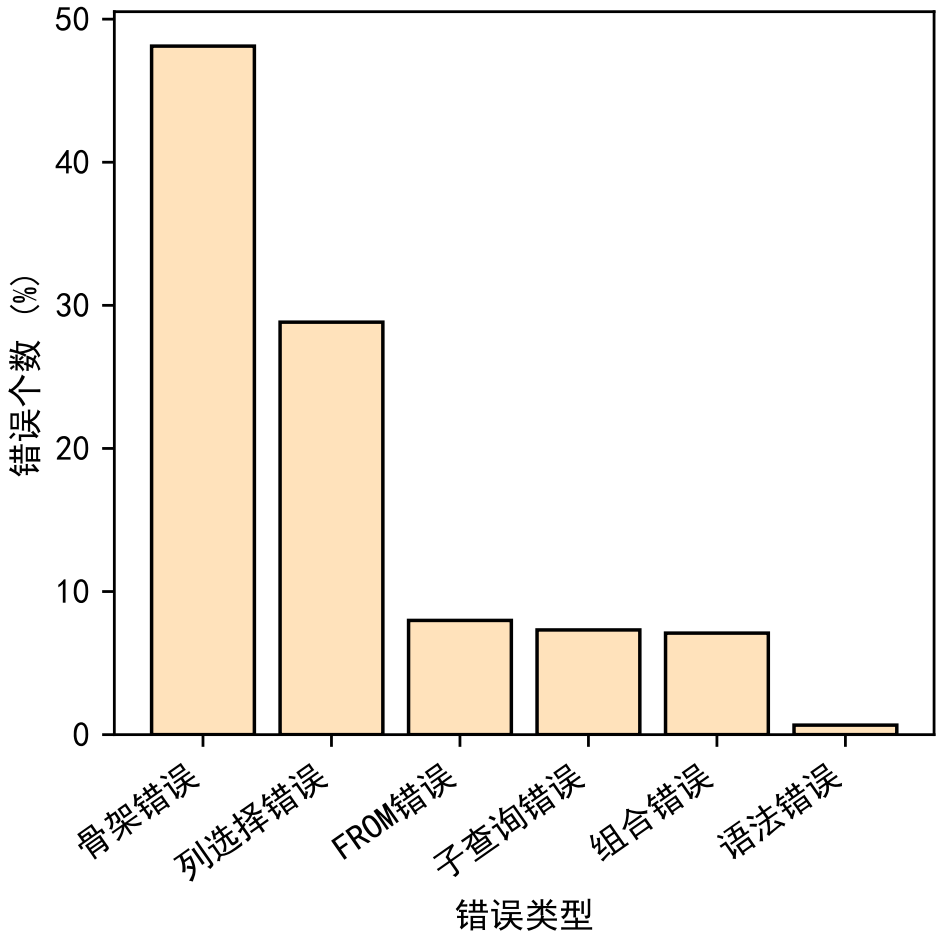

图: 预测错误分析

目录

结论

工作总结

- 提出了三项简单的改进措施，并实验验证其有效
- 融入最新预训练模型
- 在 SParC 数据集，跨领域上下文相关的自然语言到 SQL 转换任务上，取得了新的最优准确率

工作展望

- 使用类似 ORM 系统的方式，继续改进预处理方案
- 强化关系编码效果
- 数据库自然语言接口系统的进一步研究与实现