软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2022,33(11):4107−4136 [doi: 10.13328/j.cnki.jos.006539]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于自然语言的数据库查询生成研究综述*

刘喜平[1], 舒 晴[1,2], 何佳壕[1], 万常选[1], 刘德喜[1]

[1](江西财经大学 信息管理学院, 江西 南昌 330013)

[2](江西农业大学 软件学院, 江西 南昌 330013)

通信作者: 刘喜平, E-mail: lewislxp@gmail.com

摘 要: 数据库能够提供对大量数据的高效存储和访问, 然而查询数据库需要掌握数据库查询语言 SQL, 对于普通用户而言存在一定的门槛. 基于自然语言的数据库查询 (即 text-to-SQL) 在最近几年受到了广泛的关注. 对 text-to-SQL 问题的当前进展进行了系统的分析. 首先介绍了问题背景, 并对问题进行了描述; 其次, 重点分析了目前提出的 text-to-SQL 技术, 包括基于流水线的方法、基于统计学习的方法, 以及为多轮 text-to-SQL 而开发的技术, 对每种方法都进行了深入的分析和总结. 再次, 进一步讨论了 text-to-SQL 所属的语义解析 (semantic parsing) 这一领域的研究. 接着, 总结了目前研究中广泛采用的数据集和评价指标, 并从多个角度对主流模型进行了比较和分析. 最后, 总结了 text-to-SQL 任务面临的挑战, 以及未来的研究方向.

关键词: 自然语言; 数据库查询; SQL; text-to-SQL; 语义解析; 自然语言处理

中图法分类号: TP311

中文引用格式: 刘喜平, 舒晴, 何佳壕, 万常选, 刘德喜. 基于自然语言的数据库查询生成研究综述. 软件学报, 2022, 33(11): 4107–4136. http://www.jos.org.cn/1000-9825/6539.htm

英文引用格式: Liu XP, Shu Q, He JH, Wan CX, Liu DX. Survey on Generating Database Queries Based on Natural Language. Ruan Jian Xue Bao/Journal of Software, 2022, 33(11): 4107–4136 (in Chinese). http://www.jos.org.cn/1000-9825/6539.htm

Survey on Generating Database Queries Based on Natural Language

LIU Xi-Ping[1], SHU Qing[1,2], HE Jia-Hao[1], WAN Chang-Xuan[1], LIU De-Xi[1]

[1](School of Information Management, Jiangxi University of Finance and Economics, Nanchang 330013, China)

[2](School of Software, Jiangxi Agricultural University, Nanchang 330013, China)

Abstract: Database can provide efficient storage and access for massive data. However, it is nontrivial for non-experts to command database query language like SQL, which is essential for querying databases. Hence, querying databases using natural language (i.e., text-to-SQL) has received extensive attention in recent years. This study provides a holistic view of text-to-SQL technologies and elaborates on current advancements. It first introduces the background of the research and describes the research problem. Then the study focuses on the current text-to-SQL technologies, including pipeline-based methods, statistical-learning-based methods, as well as techniques developed for multi-turn text-to-SQL task. The study goes further to discuss the field of semantic parsing to which text-to-SQL belongs. Afterward, it introduces the benchmarks and evaluation metrics that are widely used in the research field. Moreover, it compares and analyzes the state-of-the-art models from multiple perspectives. Finally, the study summarizes the potential challenges for text-to-SQL task, and gives some suggestions for future research.

Key words: natural language; database query; SQL; text-to-SQL; semantic parsing; natural language processing

当前, 数据已经成为了生产要素之一, 如何有效地分析和提取数据的价值成为了人们关注的焦点. 然而, 大多数情况下对数据的分析要通过专业语言 (如 SQL) 或工具进行交互操作, 因而具有较高的门槛. 人们迫切需要新的

* 基金项目: 国家自然科学基金 (62076112, 61972184, 61762042); 江西省自然科学基金 (20192BAB207017); 江西省教育厅科技项目 (GJJ190255, GJJ180234, GJJ190208); 江西省研究生创新专项 (YC2021-B130)

收稿时间: 2021-04-02; 修改时间: 2021-06-06, 2021-09-17; 采用时间: 2021-11-15; jos 在线出版时间: 2021-12-24

方法和技术来降低数据分析的门槛, 提升数据分析的效率. 在此背景下, 基于自然语言界面的数据分析技术应运而生. 这一技术使得人们只需要使用日常的自然语言就可以完成数据分析, 而无需学习专门语言和复杂工具, 从而可以大大提升数据分析的效率和用户体验. 基于自然语言界面等新交互技术上的数据分析也已经成为业界发展的最新趋势. IT 咨询公司高德纳 (Gartner) 预计, 到 2021 年, 自然语言理解与交互式分析将促使智能商业数据分析的采用率从 35% 增加到 50%.

基于自然语言界面的数据分析的关键技术之一是基于自然语言的数据库查询技术. 数据库不能直接处理自然语言查询 (natural language query, NLQ), 需要将其转换为数据库所支持的语言, 如 SQL 查询, 这一任务在文献中一般被称为 text-to-SQL、NL2SQL (natural language to SQL) 或者数据库的自然语言界面 (natural language interface to databases, NLIDB). 图 1 是一个具体的实例, 给定图中的数据库模式, 以及自然语言查询 (如: 找出在北京降落的航班号), text-to-SQL 的任务是生成对应的 SQL 查询.

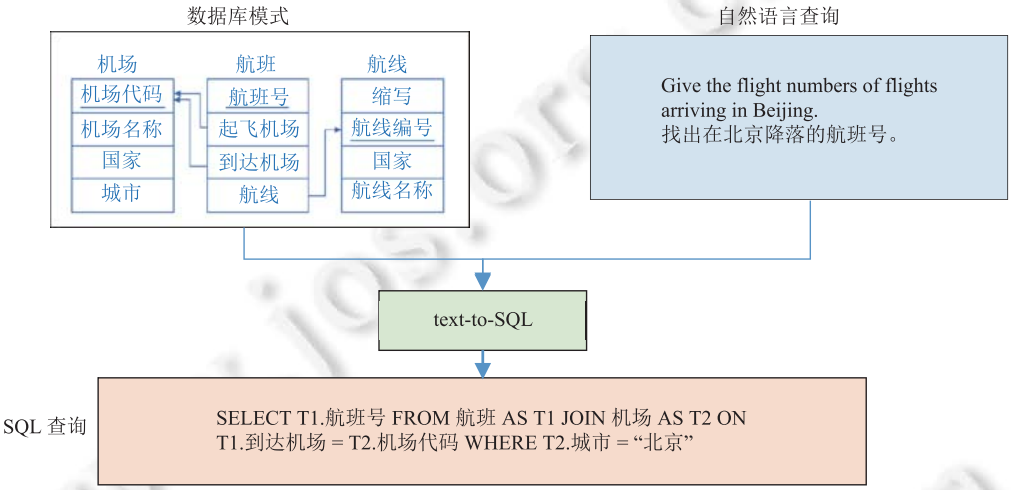


图 1 自然语言生成 SQL 查询的一个实例

text-to-SQL 问题很早就引起了研究人员的注意[1]. 早期的 text-to-SQL 方法设置了很多限制, 因而不具一般性. 如有些研究[2]为每个数据库专门创建查询语法, 而这些语法很难复制到其他数据库; 另一些研究[3,4]限定了 NLQ 的一个子集, 只有该子集中的查询可以转换为 SQL 查询, 其他的 NLQ 将无法处理. 文献 [5] 则依赖一些规则和启发式方法来生成 SQL 查询. 近几年来, 随着自然语言交互的需求不断增加, 以及深度学习和自然语言处理技术的快速发展, 越来越多的研究者关注 text-to-SQL 问题, 这些研究者既有来自于数据库领域, 也有来自于自然语言处理 (NLP) 领域. 研究者提出了一系列的新方法, 不断地推进着这一领域的研究进展.

本文对 text-to-SQL 的研究现状进行了系统的分析、比较和总结. 此外, 还对 text-to-SQL 相关的语义解析这类问题进行了介绍, 并总结了目前 text-to-SQL 研究中广泛采用的数据集、评测手段和评测结果. 最后, 对 text-to-SQL 面临的挑战和进一步的研究方向进行了总结.

1 概 述

由于自然语言和 SQL 是两种完全不同的语言, text-to-SQL 无疑是非常复杂的. 现有的 text-to-SQL 技术可以从多个角度进行分类, 如图 2 所示.

从技术路线的角度, text-to-SQL 主要有两种思路: 一是将 text-to-SQL 分解为多个阶段或者步骤, 然后每个步骤设计相应的算法来完成; 二是将 text-to-SQL 视为一个文本生成问题, 用统计学习方法直接学习一个端到端的模型. 我们把这两种思路分别称为基于流水线的方法和基于统计学习的方法.

从系统交互的角度, 可以将 text-to-SQL 任务分为单轮任务和多轮任务. 单轮任务是指只与系统进行一轮交互,

即用户提出自然语言查询, 系统直接生成 SQL; 多轮任务则要与系统进行多轮交互. 多轮任务又可以分成两种情况: (1) 非对话式: 系统生成 SQL 语句后, 获取用户的反馈以澄清提问歧义、改进生成的 SQL 和验证答案; (2) 对话式: 用户与系统进行多轮对话, 系统综合对话内容来明确用户的查询意图, 最终生成符合用户意图的 SQL 语句.

下文将首先从单轮任务出发, 对两种技术路线进行阐述, 然后再对多轮任务进行介绍.

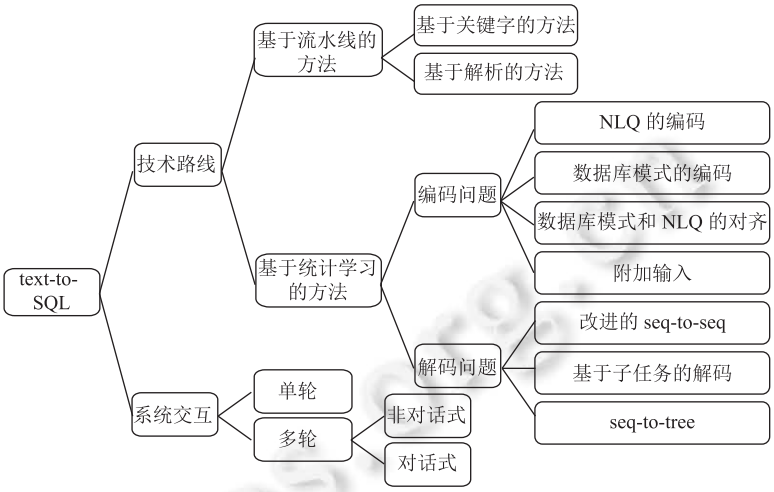


图 2 text-to-SQL 任务的分类

2 基于流水线的 text-to-SQL 方法

基于流水线 (pipeline) 的方法经过若干步将一个 NLQ 转换为 SQL, 每一步的目标和实现过程都是具体而明确的[6]. 此类方法通过词性标注、词形还原、实体识别、句法分析等 NLP 技术, 对输入的自然语言问题进行解析、化简, 识别查询中的实体及其关系, 以理解用户的查询意图, 并将自然语言查询转化为中间表达, 如关键字、某些预定义的模式或某种中间语言等, 然后再将其转换为诸如 SQL 的结构化查询语言. 由于自然语言的复杂性, 不仅单词存在多义性, 短语、句子也可能存在多种理解, 因此需要处理歧义. 不同模型处理歧义的方法包括: 与用户交互验证、对多个候选计分并进行排序, 以及引入外部知识.

一些早期的系统不支持一般的自然语言查询, 而限制用户输入仅包含关键字[7–9], 或者通过移除停用词将问题简化为关键字查询[10]. Précis[7]限制输入为通过 AND、OR 和 NOT 连接的关键字, 系统将其转换成析取范式, 然后查询数据库内容的倒排索引, 以检索每个析取项的候选解释. 多个候选解释根据模式项之间的连接权重排序, 权重可以根据模式图特征自动计算、由设计者指定或从查询日志中挖掘. SODA[8]允许用户基于关键字、运算符和值以高级语言形式提出查询, 它在数据库中的数据索引和本体中的元数据索引中查找关键字, 对生成的多个查询解释进行打分和排序, 部分执行前 10 个 SQL 查询, 为每个查询生成最多 20 个元组的结果片段供用户进行选择. SQAK[9]允许用户使用简单的关键字进行聚集查询, 该系统首先分析产生一组候选解释, 对每个候选解释构造一棵称为简单查询网络 (simple query network, SQN) 的树, SQN 可以唯一地转换为 SQL 语句, 对这些 SQN 进行打分和排序. 基于关键字的系统解决问题的思路是将数据库模式或数据元组建模为图, 与关键字进行匹配, 确定完整图上包含关键字的查询子图, 从而得出查询语句. 此类系统对用户输入的问题限制较大, 能处理的问题有限.

为了能够处理更复杂的问题, 研究者们提出了基于解析的系统, 该类系统使用句法分析等 NLP 技术, 解析输入的句子, 生成句子的结构信息, 如解析树. 解析树包含了如何将单词组合成短语, 短语组合成更复杂的短语, 直到产生完整句子的信息, 基于这些信息可以将语义映射到某些产生式规则, 因此该类系统可以更好地理解问题的语法和语义, 从而生成更准确地查询. 这类研究最具代表性的是密歇根大学开发的 NaLIX 和 NaLIR 系统. NaLIX[11]利用解析树将 NLQ 转换为针对 XML 数据库的 XQuery 查询, 通过提供反馈和改写建议, 交互式地引导用户提出

系统可理解的查询. Li 等人将该系统中的方法进一步完善, 在文献 [12,13] 中描述了一个用于关系数据库的交互式自然语言查询接口 NaLIR. 系统先将 NLQ 转换为查询树 (即依存句法树, dependency parse trees), 然后将查询树转换为 SQL 语句. 当查询存在歧义时, 系统通过查询树来进行解释, 并允许用户验证查询树.

USI Answers[14]首先使用词形还原、词性标注、命名实体识别等 NLP 技术对输入问题进行处理, 然后使用语法规则识别特定领域的术语, 使用句法规则识别引用的实体对象, 利用本体对底层数据进行建模, 用于确定查询中提到的实体之间是否存在关系, 之后根据候选实体生成不同的问题解释, 并对不同的解释进行验证和排序. TR Discover[15]使用语法规则指导和帮助用户制定问题, 这些规则使用基于特征的上下文无关文法 (feature-based context-free grammar, FCFG), 系统提供查询自动补全功能. TR Discover 首先使用 FCFG 将输入问题解析为一阶逻辑 (FOL) 形式, 然后将生成的 FOL 转换为解析树, 最后对解析树进行有序遍历, 以将其转换为可执行的 SQL 或 SPARQL 查询. Singh 等人提出的 NLTSQLC 系统[16]首先对 NLQ 进行小写转换、分词、转义词移除和词性标注, 然后将标注后的词例进一步分类为关系、属性和子句, 最后系统删除具有相同名称的歧义属性以生成最终的 SQL 表示. BioSmart[17]定义了 3 种查询类型: 迭代式、条件式或命令式, 例如, 命令式查询由动词和名词短语组成, 可以通过嵌套简单的查询类型来构建更复杂的查询. BioSmart 首先生成问题的解析树, 然后将其与每种查询类型的模板进行匹配映射, 并确定所需的表和连接路径, 最后将问题转换为 SQL 查询. Sqlizer[18]引入草图 (sketch) 作为 NLQ 和 SQL 的中间表示. 草图定义了一个 SQL 查询的骨架, 生成 SQL 的主要任务是填充其中的空槽 (slot), 这种方法可以使得生成的查询更加符合 SQL 的语法. 草图的一个示例如图 3 所示, 生成 SQL 的主要任务是预测其中的$AGG 等内容.

```
SELECT    $AGG  $COLUMN
WHERE     $COLUMN  $OP  $VALUE
(AND   $COLUMN  $OP  $VALUE)*
```

图 3 SQL 草图示例

Sqlizer 首先使用语义分析技术生成最有可能的查询草图; 对于每个查询草图, 尝试合成一个符合 SQL 语法规范、可信度高的完整形态的 SQL; 若不成功, Sqlizer 试图找出故障的原因并自动修复草图. Zheng 等人[19]提出了一种“数据+oracle”的方法来回答知识图上的 NLQ, 允许用户在理解查询时验证模糊性. 为了降低交互成本, 他们将交互问题形式化, 并设计了贪心策略来解决这个问题. Deutch 等人[20]首次研究了 NLQ 的起源 (provenance). 起源可以使答案更丰富, 对于验证答案也很有价值. 他们设计出了用自然语言表示的、以分解或概述的形式呈现起源信息的方法. 最近, 密歇根大学的 Baik 等人[6]重点关注了 NLQ 转换为 SQL 时的两个特定问题: 关键字映射和连接路径推断, 它们使用数据库的 SQL 查询日志来选择更可能的关键字映射和连接路径. nQuery[21]首先对 NLQ 进行分词和词性标注, 系统设计了数据字典用于识别子句, 然后进行表和属性的映射, 根据规则构建子句并使用过滤算法去除冗余表和属性, 最后根据 MySQL 的语法模板生成 SQL 查询. nQuery 支持聚合函数、含多个条件的 WHERE 子句、包含如“Having”和“Order by”的复杂查询. Saha 等人[22]提出的 ATHENA 系统基于本体库, 他们提出了一个两阶段方法, 即首先将输入 NLQ 转换为基于本体的中间查询语言 OQL, 然后将其转换为 SQL. 这种两阶段方法使其能够将关系存储中数据的物理布局与查询语义解耦, 从而提供物理独立性. 此外, 相对于关系模式, 本体提供了更丰富的语义信息, 例如继承和成员关系. Sen 等人在 ATHENA 的基础上提出了 ATHENA++[23], 以解决复杂的嵌套查询问题. 首先利用领域本体和语义分析进行嵌套检测, 将 NLQ 分为与各个子查询相对应的多个证据集. 然后将每个查询块的证据集翻译成 OQL, 再将这些 OQL 查询通过适当的连接条件进行连接, 以形成完整的 OQL 查询. 最后, 通过本体和数据库之间的映射将 OQL 查询转换为 SQL 查询. ATHENA++完整的流水线如图 4.

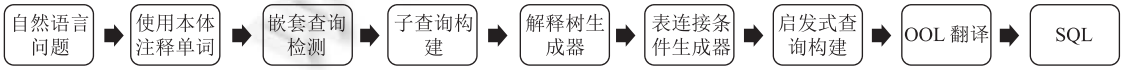


图 4 ATHENA++模型的流水线

基于 ATHENA, Sen 等人还提出了一个模式感知语义推理框架[24], 将隐式意图理解和产生语义解释的任务转化为推理任务, 用于解决跨领域的复杂 BI 查询, 将 ATHENA 的准确度提高了 30%. 该系统首先根据领域本体构建推理知识库 (reasoning knowledge base, RKB), 用于建模语义解释的规则, 例如数字聚合 (平均值、总和等) 只能应用于数值属性, 以及一些特定领域的事实 (如财务领域股票是概念而非数值属性). 对于从 NLQ 生成的初步逻辑解释, 使用本体推理器 (ontology reasoner) 检查其与 RKB 中建模规则的一致性, 不一致的部分利用更正单元进行检测和更正, 从而产生语义合理的查询解释. 例如问题“List the traders who traded more stocks than trader X (列出比 X 交易更多股票的交易者)”, 系统能够推理出“more than”适用于数值属性, 因此不能直接应用于股票类型的实体, 而股票与具有货币价值的买卖交易相关联, 其作为数值属性可以应用“more than”.

还有一些学者研究了特殊领域或生成其他结构化查询语言的自然语言查询系统. 软件项目会产生大量数据, 例如需求分析、源代码、测试文件、错误报告等, 这些数据可以存储在数据库中, 也可以存储在如 Jira 或 Github 这样的存储库中. Lin 等人提出一个自然语言界面 TiQi[25], 专门用于访问项目数据, 利用句法分析和语法规则, 除了将 SQL 和原始数据结果返回给用户外, 还生成可追溯信息模型 (traceability information model), 用于显示底层对象类和属性. 传统的关系数据库不能处理持续变化的数据, 如公交车、地铁的位置, Wang 等人提出了一个查询移动对象数据库的自然语言界面 NALMO[26], 该系统首先对 NLQ 进行预处理, 进行分词和实体识别, 然后利用位置知识库提取 NLQ 中的位置信息, 并使用前缀索引增强与位置知识库的匹配效率; 接下来系统判断查询的类型, 包括时间间隔、范围、最近邻和相似轨迹查询; 最后根据不同的查询类型, 基于规则将解析后的实体映射到相应的查询片段, 这些片段与操作符结合得到最终的结构化查询.

基于流水线的方法分析和利用问题中的语法结构和语义信息, 一定程度上可以处理复杂的输入问题并生成复杂的结构化查询. 此类方法的主要优点是不需要大量的训练数据, 可以更容易地结合领域知识, 例如许多方法使用了本体库以扩展查询, 或是引入外部词典来解决同义词问题, 如 DBpedia[27]或 WordNet[28]. 但此类方法往往对 NLQ 的表述有一定的规则限制, 或是需要手工定义一些语法规则, 因此对 NLQ 的措辞变化比较敏感. 此外, 系统中每一步的错误都可能被后续步骤放大, 例如若句法解析树错误, 即使是简单的问题系统也无法翻译[14].

3 基于统计学习的 text-to-SQL 方法

text-to-SQL 可以认为是机器翻译的一个特例. 随着近年来以深度学习为代表的统计学习方法在机器翻译领域取得空前的成功, 越来越多的研究者把目光转向统计学习方法, 不断提出新的方法和技术, 不断地刷新 text-to-SQL 的性能表现.

基于统计学习的方法依赖大量的训练数据, 其中包括数据库模式和大量的<NLQ, SQL>对, 从中学习一个模型, 实现从 NLQ 到 SQL 的转换. 这类方法一般是端到端的, 或者其中的关键部分是一个黑盒模型, 从而与流水线方法形成鲜明的对比. 目前, 基于统计学习的 text-to-SQL 方法大都采用编码器-解码器 (encoder-decoder) 框架. 这类模型的一个典型结构如图 5 所示. 在这类模型中, 编码器负责产生 NLQ 和数据库模式的表示, 解码器则根据编码器得到的表示来生成 SQL. 与一般机器翻译不同的是, text-to-SQL 的输入包括自然语言问题和数据库模式, 模型需要理解问题在数据库的哪些表和列上进行查询, 触发了哪些 SQL 操作; 此外, 生成 SQL 查询时要保证生成的语句满足语法规则并且语义正确. 因此基于统计学习的 text-to-SQL 方法需要解决两大问题: (1) 如何对 NLQ 和数据库模式等输入信息进行编码, 并将问题和数据库模式进行对齐; (2) 如何利用规则或语法指导生成 SQL. 下面分别对这两个问题及其解决技术进行介绍.

3.1 输入信息编码

3.1.1 NLQ 的编码

NLQ 编码的目的是产生一个向量表示来概括 NLQ, 该向量表示将作为解码器的输入. 各种方法对 NLQ 的编码方法比较类似. 例如, 一种典型的方法是基于 NLQ 的词向量列表, 通过双向循环神经网络 (如 Bi-LSTM, Bi-GRU 等模型) 来产生 NLQ 的编码表示, 如图 6 所示.

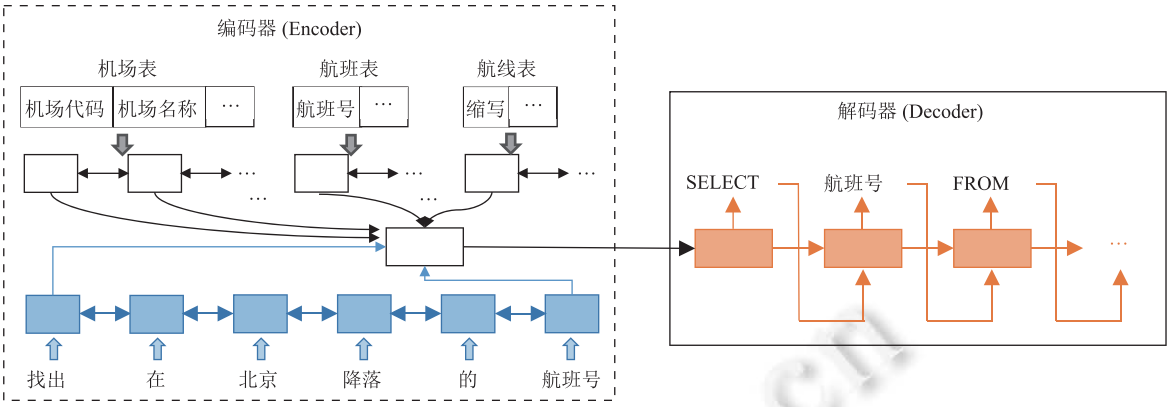


图 5 text-to-SQL 的编码器-解码器框架例子

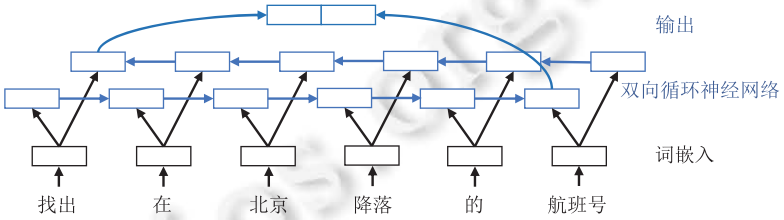


图 6 自然语言查询的编码

一些研究试图通过匿名化 (anonymization) 常量值, 即将 NLQ 中的常量值转换为特定符号来减少词汇量, 降低训练难度. NSP[29]将 NLQ 中的实体替换为它们的类型名, 对相同类型的多个实体增加编号. DBPal[30]用占位符替换 NLQ 中的常量, 使其模型独立于实际数据库. Dong 等人[31]提出了一种基于序列标注的两阶段模型对 NLQ 进行预处理, 匿名化的同时学习 NLQ 与数据库之间的语义对应关系. 该模型的第 1 阶段判断词例 (token) 是否需要匿名化, 即是否与数据库表、列或单元格相关, 第 2 阶段将词例与对应数据库模式元素进行绑定. 由于缺乏匿名化结果的标注数据, 作者提出了一种基于奖励机制的隐监督 (implicit supervision) 方法进行训练. 匿名化的结果将作为解码器的附加输入, 避免使用符号代替词例从而损害 NLQ 的语义表达能力.

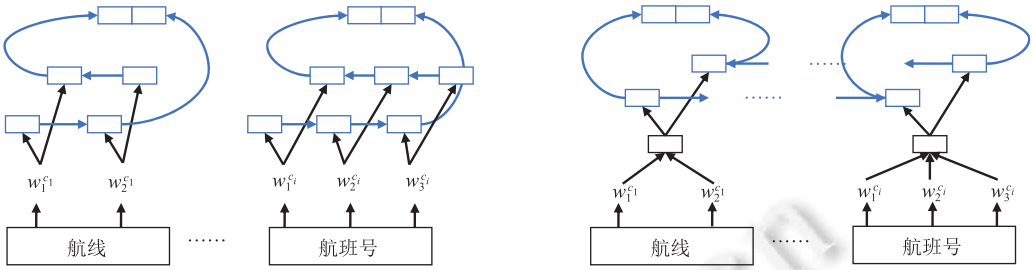
3.1.2 数据库模式的编码

正确地产生 SQL 查询需要理解数据库的模式结构. 数据库模式编码的目的也是产生一个向量表示来概括数据库的模式, 类似地, 该向量也会作为解码器的输入. 根据数据库模式的不同, 编码的对象是不同的.

(1) 单表的编码. 对单个表的编码一般有两种方法. 第 1 种是将表结构视为列名的序列, 然后用句子编码的方式来编码, 如 SQLNet[32]将列名视为单词序列, 使用 GloVe 进行词嵌入后送入 Bi-LSTM 进行编码. 第 2 种方法首先产生每个列的编码, 然后将各个列的编码拼接、累加或者再通过一个神经网络来产生表的编码. 如 TypeSQL[33]首先对组成列名的单词嵌入取平均, 以获得每个列的嵌入, 然后对表中所有列的嵌入通过 Bi-LSTM 编码. 文献 [34]中指出, 若基于数据库拆分训练集和测试集, 即一个数据库的所有问题都在同一拆分中, 则 SQLNet 的列编码方法优于 TypeSQL; 若同一数据库的问题可以同时出现在训练集和测试集中, 则结果相反. 图 7 是用这两种方法对图 1 中“航班”表进行编码的示意图.

(2) 多表的编码. 对于多表, 在编码时还需要将表名编码进来, 通常的做法是在每个列名中加入表名, 然后将所有列名按照上述方法进行编码. 最常使用的网络结构是双向循环神经网络 (如 Bi-LSTM、Bi-GRU), 也有研究将自注意力[35]应用于表的编码, 取得了不错的效果, 如 RCSQL[36]对每列将表名和列名用分隔符进行连接, 在 Bi-LSTM 的基础上通过自注意力机制形成每列的向量表示. 上述方式没有利用模式项之间预定义的一些关系, 如列与列之间的主外键关系, 而主外键关系是构建多表连接的关键, 因此 Bogin 等人[37]提出使用图来建模模式项之间的关系,

其中结点代表表和列, 边代表表和列之间的从属关系以及主外键关系, 基于图神经网络 (graph neural network, GNN) 产生每个结点即模式元素的嵌入.



(a) SQLNet 中列的编码　　(b) TypeSQL 中列的编码

图 7　单表的两种编码方式

3.1.3　NLQ 和数据库模式的对齐

生成正确的 SQL 需要将 NLQ 和数据库模式对齐, 即识别问题中词语 (实体) 是如何与数据库中的表和列对应的. 如在图 1 所示的例子中, 需要识别 NLQ 中的“航班号”对应“航班”表中的“航班号”列, “北京”对应“机场”表中的“城市”列, “降落”对应“航班”表中的“到达机场”列. “北京”是“城市”列的值, “降落”表达了“到达”的语义, 这两列都没有在问题中明确提及, 识别有一定的难度. 各个研究工作在对齐数据库和 NLQ 时主要有两种方式: (1) 分别对 NLQ 和数据库进行编码, 然后使用专门的模块进行链接和对齐; (2) 直接将 NLQ 与数据库模式统一编码, 将两者串联或建模为一个异构图.

第 1 种方式的典型代表是 IRNet[38], 其使用了一个模式链接 (schema linking) 模块, 通过字符串匹配和查询知识图谱识别 NLQ 中提到的表名和列名, 建立 NLQ 和数据库之间的链接, 增强 NLQ 和数据库模式的表示. 字符串匹配对于语义相似的同义词处理效果不佳, 如图 1 的示例中“降落”与“到达机场”的对应关系, 后续的工作更多通过注意力机制, 使得在编码 NLQ 和模式的时候给不同词条或者列以不同的注意力, 如 Ryansql[39]编码模式项时, 通过问题-列对齐层 (question-column alignment) 和问题-表对齐层 (question-table alignment), 将 NLQ 中的上下文信息添加到数据库模式的表示中. 此外, 由于 NLQ 中提到的列名或表名不一定与数据库中的名称完全相同, Ryansql 使用字符嵌入提高列名的识别能力, 利用稠密连接的卷积神经网络对 NLQ 和列中的每个单词编码, 以捕获局部信息.

对于 NLQ 与数据库模式的统一编码, 早期常用的网络结构是双向循环神经网络, 如 Seq2SQL[40]将列名、NLQ 和 SQL 关键字作为一个序列通过 Bi-LSTM 编码. 后续的工作使用 Transformer 或预训练模型, 取得了更好的效果. RAT-SQL[41]可以看作图网络 GNN[37]的后续工作, GNN 仅使用图建模了表和列的联系, RAT-SQL 在此基础上又加入表示 NLQ 的结点, 在表、列和 NLQ 三者之间定义了更多的边 (共 33 种), 部分节点和边的示例如图 8. RAT-SQL 提出了一个统一的框架, 根据模式链接以及模式内部的关系, 联合学习模式元素和 NLQ 的表示, 其关键是一种基于 Transformer[42]的关系感知自注意力机制 (relation-aware self-attention), 它既可以编码结点之间预定义的“硬”关系 (例如模式内部的从属关系、主外键关系), 同时也可以学习任意两个结点之间的“软”关系 (例如词例之间的语义关系). X-SQL[43]基于 MT-DNN[44]预训练模型, 在 NLQ 前添加 [CXT] 标记用于提取全局信息, 将 NLQ 和各列名用 [SEP] 连接成序列, 引入了一个特殊列 [EMPTY] 以处理 where 子句为空的情况, 并把原始 MT-DNN 模型中的段编码扩展为类型编码, 同时增加上下文增强层 (context enhanced schema encoder) 利用注意力机制捕获全局上下文信息. 之前的工作往往将表格中的所有列与 NLQ 拼接进行编码, 后续为了得到各列的表示需要额外的编码层或池化 (pooling) 操作, 对此 HydraNet[45]一次处理一列, 将每个列的类别、所属表名、列名信息作为文本和 NLQ 文本成对输入, 与 BERT/RoBERTa 的原始句子对训练任务完全一致. NLQ 问题中常常只提及查询条件值而省略列名或表名, 如图 1 中的“北京”省略了列名“城市”, BRIDGE[46]针对该问题利用数据库内容加强两者的对

齐. 其将问题和数据库模式串联后, 将 NLQ 和数据库内容进行模糊匹配, 匹配字段作为锚文本 (anchor text) 附加到对应的模式表示之后, 形成混合标记序列, 通过 BERT 进行编码.

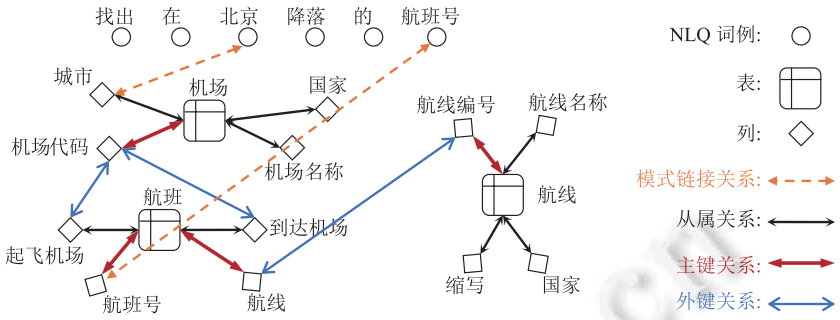


图 8 RAT-SQL 中数据库模式和 NLQ 构成的图示例

3.1.4 其他信息的编码

为了增强编码中的信息, 一些工作还在编码过程中加入了其他信息, 例如列的类型 (如数字、字符串等)、主外键约束、NLQ 中的实体、外部知识等, 这些信息有助于产生语义更丰富的表示. 增强的方式主要是将增强信息作为特征向量嵌入与词嵌入串联, 相当于对原始序列加上了标注信息. TypeSQL[33]在 NLQ 短语与数据库或知识库中的实体之间进行字符串匹配, 对于 NLQ 中每个实体, 都用相应实体的类型对其标记, 每个标记都嵌入到向量, 与单词的词向量串联作为编码器的输入. SyntaxSQLNet[47]与 TypeSQL 类似, 在对列编码时加上了列的类型, 包括字符串、数字、主键和外键. Cai 等人[48]将新的语义特征输入到编码器中, 如 NLQ 中的值/字符串表达式、提到的数据库列和表的单词. 这些附加特征帮助神经网络更好地专注于理解自然语言中的操作语义, 并减轻了 SQL 语法学习的工作量. Liu 等人[49]将列值的极性作为外部知识显式地合并到输入信息的编码中, 从而更好地理解列与其值之间的比较关系, 这一信息能够更准确地产生比较关系条件. 文献 [50] 提出了一种基于数据表内容的字段嵌入方法, 利用数据表内容增强 NLQ 和数据库模式的语义表示. Wang 等人[51]提出一种神经网络分类器用于检测 NLQ 中的列名和值, 利用对抗文本机制定位它们的位置, 将 NLQ 中这些特定于数据的部分进行注释后联合数据库模式编码, 使得模型更好地理解 NLQ 的潜在语义结构.

3.2 SQL 生成

text-to-SQL 本质上是一种文本生成任务, 但是与一般的文本生成任务 (如机器翻译) 相比, text-to-SQL 具有以下特点: (1) SQL 有严格的语法和语义要求, 如 FROM 子句不能为空, SELECT 子句中的列名必须来自于 FROM 子句中的某一个表, 连接条件必须正确等; (2) SQL 中的部分文本之间顺序不重要, 如 SELECT 子句中的列名、WHERE 子句中的部分条件可以交换顺序, FROM 子句的写法也不唯一. 这些特点使得 SQL 既严格又灵活, 为 SQL 生成增加了很多困难, 使得 SQL 生成不能直接照搬机器翻译等领域的 seq-to-seq 方法. 现有的 SQL 生成方法可以总结如下.

3.2.1 改进的 seq-to-seq 方法

这类方法基本采用 seq-to-seq 解码方法, 但是针对 SQL 的特点进行了改进. 在传统的 seq-to-seq 模型中, 解码器往往使用 softmax 分类器在预定义的词表中预测单词, 但无法处理未登录词 (out of vocabulary, OOV). 在 text-to-SQL 任务中, 若根据训练数据构建词表, 在新数据上可能面临大量未登录词. 传统机器翻译方法使用的词表很庞大, 当生成任务的目标语言是 SQL 时, 其语法的符号有限. 基于以上两点, 许多基于 seq-to-seq 的工作使用或改进指针网络 (pointer network)[52], 将 SQL 关键字、NLQ 中的单词和数据库元素作为输入序列, 利用注意力机制, 在解码过程中使用“拷贝”机制, 从输入序列中选择每一步生成的单词, 以解决 OOV 问题[53], 并减少预测空间的大小.

Wang 等人[54]使用一个 seq-to-seq 模型, 根据输出类型, 解码器或者使用基于注意力的复制机制从 NLQ 中复制输出符号, 或者从固定词汇表中生成输出符号. 文献 [55] 同样对指针网络进行扩展, 不同的是其分别对列名和

问题进行编码. 由于自然语言表达的多变性, 问句中的表述可能与列名或单元格内容不一致, 导致指针网络生成的 SQL 包含了不能执行的结果, 对此 Sun 等人[56]提出了融合 SQL 语法的生成式语义解析模型 STAMP, 在指针网络的基础上进行改进. STAMP 的解码器由 3 个频道和 1 个门单元组成. 3 个频道分别为 Column、Value、SQL 频道, 分别预测列名、表中单元格值和 SQL 关键字, 而门单元则预测在每个时间节点应该选择哪个频道的预测结果来输出. Wang 等人[57]首先使用基于 seq-to-seq 的模型将关于表内容的问题转换为 SQL 查询, 然后使用注意力-复制元算法编辑生成的查询, 使用特定于任务的查询表进一步编辑查询. BRIDGE[46]将训练集中每个 SQL 语句的子句按标准数据库执行顺序重新排列, 根据重排之后的语法约束启发式修剪解码器的搜索空间, 其解码器是结合多头注意力机制的指针网络.

3.2.2 基于子任务的解码方法

一些研究在设计解码器时将 SQL 分解为不同的部分, 有针对性地设计不同的子任务, 修剪生成查询的空间, 简化生成问题. Seq2SQL[40]的模型分为 3 个组件: (1) 聚合分类器: 选择聚合类型, 如 sum、count 等; (2) select 列解码器: 基于指针网络选择要查询的列; (3) where 条件解码器: 基于指针网络选择条件语句. 但 Seq2SQL 存在顺序问题, 因为 SQL 语法中某些子句中的词条之间并没有顺序要求, 后续工作对此做出了改进. SQLNet[32]将 SQL 生成解耦为 6 个子任务, 包括选择列的 Select-Column、选择聚合函数的 Select-Aggregation、预测条件个数的 Where-Number、预测条件列的 Where-Column、选择条件操作符的 Where-Operator, 以及预测条件值的 Where-Value. SQLNet 将草图 (sketch) 引入了 SQL 生成过程. 在生成 WHERE 子句时, SQLNet 首先预测列的集合, 然后针对每个列, 根据依赖关系预测操作符和条件值, 避免了 WHERE 子句中条件的顺序问题. 许多后续方法也使用了草图来描述 SQL 查询. Dong 等人[58]提出了一种从粗粒度到细粒度的解码框架, 将 SELECT 子句中的列和聚集函数的预测视为分类任务; 对于 WHERE 子句, 则首先使用分类模块生成粗略草图, 确定条件数和条件运算符, 再根据草图预测条件列和条件值. Guo 等人[59]和 SQLova[60]继承了 SQLNet 任务解耦的思路, 不同之处在于, Guo 等人采用了双向注意力机制和卷积神经网络 (CNN) 进行字符级嵌入来改善结果, SQLova 使用了 BERT 作为模型的输入表达层, 代替了词向量. TypeSQL[33]将基于知识的类型感知文本引入到 SQL 生成器中, 通过合理地对不同的槽进行分组并捕获属性之间的关系, 与 SQLNet 相比训练时间减半且效果提高了约 3.5%. X-SQL[43]也将 SQL 语句的生成分解为 6 个子任务, 与其他研究不同的是, 其他研究在生成 WHERE 子句时使用了多个二分类器, 每个分类器独立优化且它们的输出不能直接比较, 不能有效建模列之间的关系; 而 X-SQL 提出一种基于 KL 散度的列表式全局排序方法 (list-wise global ranking) 作为新的目标函数, 优化各个子任务的损失总和, 取得了更高的准确率.

上述工作均只考虑单表上的简单查询, 其中查询仅包含 SELECT 和 WHERE 子句, 且 SELECT 子句中仅包含单个表达式. HydraNet[45]针对的是多个 SELECT 和 WHERE 子句的多表非嵌套查询, 由于其编码器基于 BERT 将每列信息和 NLQ 成对输入, 与特定列相关的子任务 (如聚合运算符) 被建模为句对分类任务和问答任务; 与特定列无关的全局子任务 (如 WHERE 条件的数量) 通过混合排序机制综合所有列的结果. M-SQL[61]对 X-SQL 进行了改进. M-SQL 包含 8 个子任务, 合并了 X-SQL 中的 W-NUM 与 W-OP 模型, 增加了 3 个子模型: S-NUM, 用于对 SELECT 字段数进行预测; W-COL-VAL, 抽取查询中的所有值; W-VAL-MATCH, 对所抽取的值和表字段进行匹配. F-SQL[62]在 M-SQL 的基础上, 使用门机制融合数据库模式和表内容, 有效地增强了列表示, 提高了 SQL 查询生成性能.

还有一些研究工作特别针对 Spider 数据集上的复杂查询任务. 文献 [63] 提出了一种两阶段方法, 首先生成 SQL 子句, 将其分为 4 个子任务并建模为多个多分类问题; 然后根据子句提及的表和数据库模式图, 采用斯坦纳树 (Steiner tree) 生成算法生成 FROM 子句中多表连接路径. RCSQL[36]提出了一种基于子句的解码模型. 该模型分为草图生成和槽填充两大模块, 首先使用文本分类模块为每个 SQL 子句预测一个草图, 如 SELECT 子句的草图为 "($AGG $COLUMN)*", WHERE 子句的草图为"$AGG $COLUMN $OP VALUE(AND/OR $AGG $COLUMN $OP VALUE)*", 然后根据草图查找列和相应的运算符. RCSQL 为每个 SQL 子句单独训练一个特定于子句的解码器, 每个子句最多由 3 种不同类型的模块 (草图、列、运算符) 结合来预测. 当模型预测到嵌套查询, 暂时将 [SUB QUERY] 标记插入相应位置, 并将带标记的 SQL 作为附加输入, 递归地生成嵌套查询. Ryansql[39]也分为草

图生成和槽填充两阶段, 但不同的是 RCSQL 将草图生成拆分为 8 个子任务, Ryansql 拆分为 11 个. 此外对于嵌套语句, Ryansql 将 SQL 语句拆分为非嵌套 SELECT 语句块, 引入语句位置编码 (statement position code, SPC) 定义各个语句块间的关系, 然后递归地预测 SPC 及对应 SELECT 语句块.

3.2.3 seq-to-tree 的方法

不受约束的解码器产生的输出序列很可能存在语法错误, 无法执行, 尤其是复杂问题对应的 SQL 查询. seq-to-tree 的方法生成 SQL 查询的语法树, 使用语法约束解码器生成有效的输出, 保证生成的 SQL 查询具有正确的句法和语义规则. Yu 等人提出的 SyntaxSQLNet[47]将 SQL 语句分解为 9 个模块, 每个模块对应了 SQL 语句中的一种成分. 其解码器结合 SQL 生成路径历史预测 SQL 语法树, 解码时由预定义的 SQL 文法确定 9 个模块的生成顺序, 从而引入结构信息. 基于树的解码器使 SyntaxSQLNet 能够根据 SQL 语法递归地调用不同的模块, 系统地生成嵌套查询, 如图 9 所示, “ROOT”代表查询的开始, 系统首先调用“IUEN”模块预测是否存在 INTERSECT、UNION 或 EXCEPT, “NONE”表示不存在, 之后调用“KW”模块生成 WHERE 和 SELECT 关键字, 以此类推, 依次调用对应模块生成相应的内容, “none”表示没有聚集函数. SyntaxSQLNet 根据 SELECT 语句中列所属的表构成 FROM 子句, 不进行预测, 因此图中的树没有 FROM 子句.

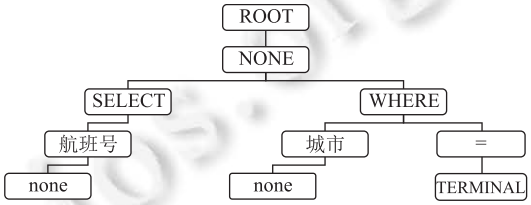


图 9 SyntaxSQLNet 中基于树的 SQL 生成

SemtoSql+ [64]根据语义层次将查询分为 4 个级别, 依据 SQL 子句构建特定语法树的解码器. Lin 等人[65]研究了基于语法生成抽象语法树 (abstract syntax tree, AST) 的模型, 通过设计浅层解析表达式语法使得语法树更加容易学习. 他们首先设计了一个公共的 SQL 语法基础, 对于那些特定于数据集的部分, 通过在语法中增加一些其他非终结符来处理. 基于 Lin 等人提出的语法规则, Bogin 等人[37]基于 GNN 产生模式元素的嵌入, 在解码时利用自注意力机制考虑模式信息, 使用语法规则扩展非终结符, 生成 SQL 语法树. RAT-SQL[41]同样通过生成 AST 来推断最终的 SQL 查询, AST 通过预测动作来构造, 动作包括 APPLYRULE、SELECTTABLE 和 SELECTCOLUMN, 其中 APPLYRULE 应用语法规则扩展非终节点, 生成 SQL 基本结构; SELECTTABLE 或 SELECTCOLUMN 选择相应的表和列作为叶结点.

3.2.4 其他方法

除了上述方法外, 研究者还提出了基于强化学习的方法. Seq2SQL[40]提出用强化学习来生成执行正确的查询语句, 由于同一个 SQL 查询可能具有多个等效的序列形式, Seq2SQL 在解码器生成任何等效查询时对解码器进行奖励. 谷歌的研究者提出了一种基于弱监督强化学习的方法 MAPO[66]来解决 text-to-SQL 任务. 在 MAPO 中, 强化学习的状态是输入的 NLQ 和其对应的环境 (解释器或者数据库), 动作空间是当前 NLQ 下所有可能产生的程序集合, 而每一条强化学习轨迹的动作序列对应着每一个可能的程序. MAPO 算法的目的是生成一个策略函数, 该函数给出了在一个 NLQ 条件下, 采样到的各个程序的概率分布, 根据此概率分布, 可以得到 NLQ 对应的程序. MAPO 存在稀疏回报和虚假的高回报问题, 对此文献 [67] 基于 MAPO 提出了其改进方案 MAPOX, 用于解决在稀疏回报情况下高效探索样本空间的问题, 并提出了 MeRL (meta reward-learning) 用来处理虚假的高回报样本程序.

一些工作使用重排序算法, 在输出束 (output beam) 里选择最优的预测来提升准确率. Bogin 等人[68]在 GNN 模型[37]的基础上提出了一个语义解析器, 对数据库结构和问题进行推理, 基于全局信息进行重排序, 从而对查询中应使用哪些数据库常量做出全局决策. Bertrand-DR[69]提出了一种判别式重排序器, 将重排序器构建为与模式无

关的 BERT 微调分类器, 使得 GNN 模型[37]在 Spider 数据集上的准确度相对提高了 12.9%.

文献 [70] 引入了一种称为执行指导 (execution guidance) 的新机制以利用 SQL 的语义. 它以部分生成的程序的执行情况为条件, 在解码过程中检测并排除故障程序. SQLova[60]、BRIDGE[46]、X-SQL[43]和 HydraNet[45]使用执行指导机制后, 均取得了更好的结果. 考虑到通常存在多个具有相同或非常相似语义的正确 SQL 查询, IncSQL[71]提出使用非确定性预言机 (oracle) 训练序列到动作模型, 用若干个动作之一逐步填充 SQL 查询的槽, 结合执行指导机制和束搜索, 解码器保留得分最高的前 k 个部分 SQL 查询.

3.3 其他研究问题

基于统计学习的方法, 尤其是基于深度学习的方法, 不需要特征工程, 在对 NLQ 变化的鲁棒性方面显示出令人鼓舞的结果, 但其需要大规模高质量的训练数据, 限制了它们在新领域中的使用, 因此一些工作研究少样本 text-to-SQL 和数据增强等问题, 以期在有限的样本条件下提升模型的效果.

(1) 少样本 text-to-SQL

Huang 等人[72]借助于域依赖的相关函数, 通过将原始学习问题简化为 few-shot 元学习场景, 与非元学习方法相比, 可以实现 1.1%–5.4% 的绝对准确度提升. Lee 等人[73]提出了一个 one-shot 学习模型, 该模型能够基于一个样本生成未经训练的 SQL 模板. 该模型分为两个阶段: SQL 模板分类和基于指针网络的槽填充. 对于模板分类, 该文提出了候选搜索网络, 从整个模板中选择排名靠前的 n 个最相关的 SQL 模板以构建支持集. 随后, 应用匹配网络 (matching network) 根据支持集对模板进行分类. Chang 等人[74]设计了一个简单但有效的辅助任务用于零样本 (zero-shot) text-to-SQL 任务, 显式地建模从自然语言实体到数据库模式元素的映射. 该任务可作为支持模型以及生成任务的正则化项, 以增加模型的泛化能力.

(2) text-to-SQL 中的数据增强

还有一些工作研究了数据增强技术. SyntaxSQLNet[47]首先在 Spider 数据集上提取通用的 NLQ-SQL 模板, 剔除简单模板后得到 280 个复杂度较高的模板, 在 WikiSQL 数据集上对每张表随机使用 10 个模板进行填充, 填充时保证数据类型的一致性, 共产生了约 98 000 个样本用于训练. Guo 等人[75]提出了一个基于问题生成的语义分析训练框架, 首先使用一个基于规则的 SQL 采样器生成 SQL 语句, 随后用一个在小规模有监督数据上训练的问题生成模型, 生成多个高置信度的问题, 将新生成的数据与小规模的有指导数据结合, 共同训练语义分析模型. Zhong 等人提出了 GAZP[76], 使用前向语义解析器和后向话语生成器来合成数据, 使现有的语义分析器适应新的数据库环境. 针对新的数据库, GAZP 首先基于语法对 SQL 逻辑形式进行采样, 然后使用生成器生成与这些逻辑形式相对应的句子, 再使用解析器解析生成的句子, 保留解析结果与原始逻辑形式等效的数据. 最后, 原始训练数据与符合一致性的合成数据共同用于训练语义解析器.

(3) 面向 text-to-SQL 的预训练模型

预训练语言模型从大规模语料中学习语言的通用表示, 大大提高了模型的准确率, 然而这些预训练模型基本是在文本上进行训练的, 对结构化的表格数据进行编码时, 会存在形式不一致的问题, 因此近期一些工作有针对性地提出了新的预训练模型. Yin 等人[77]提出了预训练语言模型 TaBert, 以 2 600 万张表格和与其相邻的文本作为训练数据, 共同学习 NLQ 和结构化表格的表示. 为了处理大型表格, TaBert 将与描述最相关的数据库子集作为内容快照, 将其中的每一行线性化, 使其与基于 Transformer 的 BERT 模型兼容, 并结合垂直自注意力机制, 聚合不同行单元格之间的信息, 最终经过池化层得到单词和列的表示. TaBert 为学习自然语言和列的上下文表示设计了不同的预训练任务, 前者使用标准的掩码语言模型 (masked language model, MLM), 遮蔽句子中 15% 的词例进行恢复, 后者使用遮蔽列预测 (masked column prediction, MCP) 和单元格值恢复 (cell value recovery, CVR) 任务, 在行线性化的过程中随机遮蔽 20% 的列名称及数据类型, 保留其单元格值, 训练模型预测列名及其类型, 并通过单元格的向量表示恢复单元格的原始值. TaPas[78]扩展了 BERT 的结构, 从维基百科中提取了 620 万条表格和文本片段进行预训练, 将表格平铺成单词序列连接到 NLQ 之后, 添加了额外的列 ID 嵌入、行 ID 嵌入等用于编码表的结构, 还添加了序数嵌入 (rank embedding) 编码列值的顺序. TaPas 使用掩码语言模型作为预训练目标, 遮蔽文本和表格中

的一些词例进行恢复. TaPas 使用两个分类层, 用于选择单元格和聚合操作符, 不生成逻辑表达式, 直接通过表格回答问题. GraPPa[79]认为 TaBert 对大量的噪声数据进行预训练速度慢且代价高昂, 其从现有 text-to-SQL 数据集中导出同步上下文无关文法 (SCFG), 在高质量表格上自动生成问题和 SQL 对. 在标准 MLM 任务的基础上, GraPPa 增加了一种新的“文本模式链接”目标对合成数据进行预训练, 该目标预测列是否出现在 SQL 查询中以及触发哪些操作, 用于捕获特定于表格数据的语义. GRAPPA 在 4 个语义解析数据集上都取得了最优结果.

(4) 多任务学习

文献 [80] 介绍了自然语言十项全能 (decaNLP) 任务, 该任务涵盖了问答、机器翻译、摘要、语义解析等 10 项 NLP 任务, 并提出了一个新的多任务问答网络 (MQAN), 该网络可以联合学习 decaNLP 中的所有任务, 而无需任何特定于任务的模块或参数. 实验结果显示 MQAN 在单任务设置中的 WikiSQL 语义解析任务上获得了最先进的结果.

综上所述, 基于统计学习的方法不需要手动定义转换 SQL 的规则, 对自然语言的变化更健壮, 具有更大的灵活性, 在 text-to-SQL 任务中取得了较好的效果, 表 1 对两种技术路线进行了总结和比较.

表 1　两种技术路线的比较

方法	核心思想	优势	不足
基于流水线的方法	借助中间表达 通过多个步骤解决问题	过程清晰 容易结合用户反馈和领域知识 解释性强 无需大量训练数据	对NLQ的表述有一定的规则限制 需要手工定义语法规则 多步之间误差传递
基于统计学习的方法	编码器-解码器框架	端到端 学习能力和泛化能力强 对NLQ表述变化鲁棒性高	依赖大量标注数据 可解释性差 训练难度大

4　多轮 text-to-SQL

上述工作都是考虑单轮对话, 即用户给出一个问题, 系统一次性生成对应的 SQL 查询. 在实际应用中, 这种场景存在一些问题: 一是用户往往不能一次性准确表达出查询, 二是系统往往很难一次性生成满意的查询, 三是用户很难评估生成的 SQL 是否正确. 解决这些问题有一个共同的思路, 那就是增加用户和系统的交互, 通过交互来完善用户查询意图, 改进生成质量. 基于这一思路, 就产生了多轮 text-to-SQL 任务. 根据生成 SQL 时是否考虑交互上下文, 多轮 text-to-SQL 又可以分为基于反馈的多轮 text-to-SQL 和上下文感知的多轮 text-to-SQL.

4.1　基于反馈的多轮 text-to-SQL

基于反馈的多轮 text-to-SQL 中系统交互的目的是获取用户反馈, 通过反馈改进模型, 或者是促使用户理解查询结果并引导用户修正问题, 在生成 SQL 的时候不考虑对话上下文, 其框架如图 10 所示.

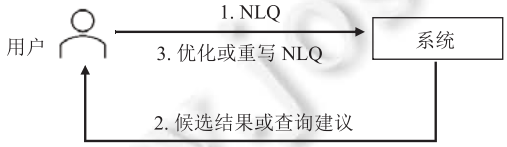


图 10　交互模型框架

文献 [29] 提出了一种简单的交互式学习算法, 首先部署初步的模型, 然后使用用户反馈和选择性查询注释来迭代地改进模型. 算法在“训练模型”和“预测并收集反馈”之间交替. 文献 [81] 设计一个对话系统, 通过多次询问用户多项选择问题来提取和验证生成查询中的潜在错误. Duoquest[82]系统提出了双重规范查询合成 (dual-specification query synthesis), 使用 NLQ 和可选的类似实例编程 (programming-by-example, PBE) 的表格草图查询 (table sketch query, TSQ) 作为输入, TSQ 允许用户指定查询约束, 如列的数据类型, 查询结果必须满足 TSQ. 该模

型使用户能够表达领域知识, 并有效修剪查询的搜索空间. 如果候选查询结果不满足用户需求, 用户可以改写NLQ 或向 TSQ 添加更多信息, 直到用户获得所需的查询为止. Li 等人[83]提出了一种独立于模型的交互式方法, 该方法定位 NLQ 中模型难以理解的词, 生成选择题与用户进行交互, 根据用户的反馈重写 NLQ, 可以在有限的交互时间内增强模型的性能. 文献 [84] 提出了一个根据自然语言反馈进行 SQL 校正的任务, 针对该任务构建了数据集 SPLASH, 并创建了一个用自然语言解释 SQL 的框架, 使不精通 SQL 的用户能够理解 SQL 查询, 进而提供反馈以更正不正确的 SQL.

4.2 上下文相关的多轮 SQL 生成

最近一些工作考虑更复杂的上下文相关的应用场景. 在实际应用中, 用户经常通过询问一系列相关问题与系统进行多轮交互来获取信息. 随着交互的进行, 用户通常会参考历史记录中提及的相关内容, 或者假设系统已知而省略先前传递的信息. 因此, 在依赖于上下文的场景中, 上下文历史对于理解用户的后续问题至关重要, 并且系统经常需要重现先前回合中生成的部分序列. Suhr 等人[85]提出了一个上下文相关的模型将交互中的话语映射到可执行的形式查询. 为了合并交互历史, 该模型维护了一个交互级的编码器, 该编码器在每个回合后更新, 并且可以在生成期间复制先前预测查询的子序列. SyntaxSQL-con[86]通过为解码器提供前一个问题的编码作为附加上下文信息来扩展 SyntaxSQLNet[47]. Zhang 等人[87]研究了一种基于编辑的跨域上下文相关 text-to-SQL 生成方法 EditSQL, 如图 11. EditSQL 对 NLQ、数据库模式和历史信息交互编码, 解码时考虑通过对前一轮查询应用复制和编辑操作来生成当前查询. 为此, 将前一轮生成的 SQL 查询视为一个词例序列, 然后在生成新的查询时在词例级别重用先前的生成结果.

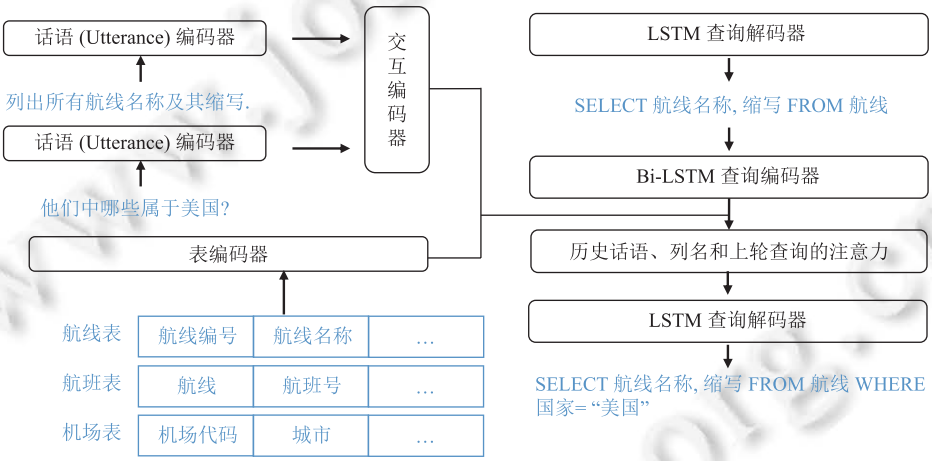


图 11 EditSQL 的模型架构

EditSQL[87]仅考虑了用户的历史输入和上轮预测的查询, 忽略了数据库模式项的历史信息. 在此基础上, IGSQL[88]提出了一种数据库模式交互图编码器, 基于数据库模式构造图, 利用注意力机制, 使用前一轮和当前轮中相邻的模式项更新当前轮的模式项表示, 对会话场景中的数据库模式项进行建模, 保证数据库语义的一致性. 此外, 在解码阶段, IGSQL 引入门机制来权衡 SQL 保留字、数据库模式项和之前轮预测的 SQL 词例的重要性. IST-SQL[89]受面向任务的对话系统中对话状态跟踪组件的启发, 分别基于模式项和 SQL 关键字定义了两种交互状态, 并设计了一个关系图神经网络和一个非线性层来分别更新这两个状态的表示. 这些状态表示与话语表示、上轮预测的 SQL 结果相结合, 利用复制机制对当前轮的 SQL 查询进行解码. Liu 等人[90]将多轮问题中的上下文总结为 3 种: 语义完备 (即上下文无关)、共指和省略, 作者在同一个解码器上评估了 13 种不同的上下文编码方法, 其中包括针对问题上下文及 SQL 逻辑形式上下文的各 3 种方法, 以及它们的 7 种组合方法. 该文同时提出了一种基于语法解码的语义解析器, 输出与 SQL 查询的抽象语法树对应的语法规则序列. Hui 等人[91]提出了一个动态图框架, 图网络中的节点、边和关系权重的表示会在交互式上下文中进行调整, 使得该框架能在对话进行时有效地建模上

下文话语、词例、数据库模式及其复杂的交互. 框架采用动态记忆衰减机制结合归纳偏好, 鼓励模型忘记较长历史的部分动态图, 同时在话语级别和词例级别上集成丰富的上下文关系表示, 并通过重排序模块进一步改善结果.

5 从 text-to-SQL 到语义解析

text-to-SQL 的本质是将用户的自然语言语句转化为计算机可以理解并执行的规范语义表示, 是语义解析 (semantic parsing) 领域的一个特例. 语义表示除了 SQL, 还有逻辑形式和可执行程序, 如 lambda-表达式[92]、语义图[93]、机器人控制指令[94]等表示形式, 如图 12 所示. 语义解析具有广泛应用, 如基于知识库的问答 (KBQA)[95]、场景中的物体识别[96]、机器人控制[97]等. 本节介绍 text-to-SQL 之外的语义解析的研究.

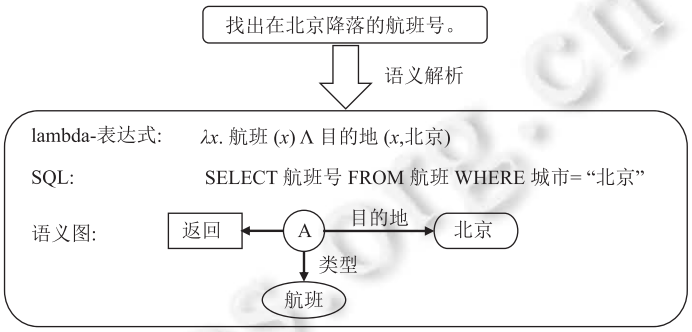


图 12 语义解析任务中语义表示的不同形式

(1) 基于词典-组合文法的语义解析

早期的语义解析基于领域规则, 开发这些系统需要大量特定领域的知识, 人工构建规则, 因此鲁棒性差, 难以迁移到其他领域, 其中一个代表是 LUNAR 系统[98]. 机器学习兴起后基于统计的方法逐步代替了规则方法, 其中词典-组合文法的语义解析是典型代表. 此类方法使用词典进行词汇映射, 识别基本语义单元, 基于文法规则组合语义单元, 产生候选解析结果, 再由机器学习的评分模型进行打分. 经典的组合文法包括组合范畴文法 (CCG)[99]和基于依存的组合语义文法 (DCS)[100]. 后续许多工作专注于语法规则的改进或学习质量更高的词典. 文献 [101] 使用松弛的 CCG 文法, 通过放松语法的某些部分, 如允许词序变化、允许省略等, 使得模型适应自然语言的变化. Kwiatkowski 等人[102]通过引入因子化词典, 将词汇分解为词素以解决词典稀疏、难以学习的问题. 基于词典-组合文法的方法过程清晰, 可解释性强, 但需要学习词典和定义文法, 依赖于词典和文法的覆盖度.

(2) 基于神经网络的语义解析

最近, 深度学习方法也在语义解析领域逐渐流行开来. 一种常见的思路是将语义解析为视为 seq-to-seq 的机器翻译问题, 使用基于循环神经网络的编码器-解码器框架. 通常语义解析的解码结果是结构化的, 直接使用序列解码器无法表达语法和结构方面的限制, 因此许多工作研究了受约束的解码器. Jia 等人[103]使用基于注意力的复制机制处理 OOV 问题, 并引入了一种数据增强方法, 利用同步上下文文法生成额外的训练数据. Dong 等人[104]提出一种分层的树形 LSTM 解码器, 在输出词表中增加了非终节点, 若预测结果为非终节点, 则将其状态作为输入预测下一层的子序列, 以此类推直到生成叶节点为止. 该方法能够保证生成的词例符号匹配正确, 但不能保证生成的结果可执行, 可能会生成结构有效但无意义的逻辑表达式. 利用语法对解码过程进行限制, 能够保证生成的代码或查询是可执行的. Xiao 等人[105]将语法约束融合到基于 RNN 的序列预测. Rabinovich 等人[106]引入了抽象语法网络 (ASN), 利用模块化解码器, 以自上而下的方式生成抽象语法树. Yin 等人[107]提出了一种语法驱动的神经代码生成模型, 该模型将抽象语法树的生成形式化为应用动作的序列, 这些动作要么应用生成规则, 要么发出终端符号. 因此, 程序设计语言的基础语法在语法模型中先验地编码为可能动作的集合. Krishnamurthy 等人[108]使用实体嵌入和链接模块, 将问题词例链接到它们所引用的实体, 其解码器的动作空间由一个类型受限的语法来定义, 保证生成的逻辑形式满足类型约束. Chen 等人[109]设计了一个序列到动作的模型来生成语义图, 解码过程中加入结构约

束和语义约束, 以确保预测形成连通无环图, 并遵循从知识库提取的特定域的语义约束. Dong 等人[58]将语义解析过程分解为两个阶段. 首先, 给定输入的句子, 生成描述其含义的粗略草图, 它会掩盖低级信息 (例如变量名和参数); 然后, 以自然语言和草图为输入填充缺少的细节. 粗略草图有一定的泛化性能, 先生成粗略草图要更容易, 且可以为后续的解码提供指导.

基于神经网络的方法需要大量标注数据, 代价较高, 一些工作尝试使用弱监督学习、迁移学习、预训练模型等降低数据标注的成本. 文献 [110,111] 利用语义表示的执行结果, 如问题的答案来监督模型的学习; 文献 [112] 使用了大量的网络文本和知识库中实体对的共现信息来建立短语和谓语之间的对齐, 并提出了一种桥接操作自动引入二元谓词, 扩充词典; Berant 等人[113]提出了一种基于复述的方法利用知识库未覆盖的文本. Herzig 等人[114]根据语言结构在不同领域下具有通用性的特点, 解耦结构和词典, 使用包括 4 个独立模块的流水线框架实现零样本语义解析. Sun 等人[115]利用规则覆盖的数量有限的样本作为初始化, 在没有自然语言对应的逻辑表达式或程序执行结果的场景下, 在反向翻译范式中使用语义解析器和问题生成器, 生成问题程序对进行迭代训练, 构建神经语义解析器. 文献 [116] 研究了弱监督与主动学习之间的交集, 使用弱监督方法训练语义解析器, 然后采用启发式的数据选取方法, 选取一部分训练数据, 主动寻求额外的标注, 以改进在弱监督下训练的模型. Shao 等人[117]提出了一个多语种语义解析模型, 该模型首先使用多级对齐机制进行预训练, 使用对抗训练方法来对齐不同语言的空间, 并使用句子级和单词级平行语料库作为监督信息来对齐不同语言的语义, 然后再共同训练多语言语义解析和多级对齐任务.

6 数据集与评价方法

本节介绍 text-to-SQL 任务的常用数据集和评价方法.

6.1 数据集

无论是 text-to-SQL 还是语义解析, 都需要大量的训练数据. 研究人员提出了多个 text-to-SQL 标注数据集, 极大地推进了这一领域的研究.

1) ATIS[118,119]. ATIS 来源于机票订阅系统, 由用户提问生成 SQL 语句, 是一个单一领域且上下文相关的数据集. 包含 27 张表以及不到 2 000 次的问询, 每次问询平均 7 轮, 93% 的情况下需要联合 3 张以上的表才能得到答案, 问询的内容涵盖了航班、费用、城市、地面服务等信息. ATIS 中存在嵌套查询, 仅有 5 个样本含 GROUP BY 语句, 不含 Order By 语句[34].

2) GeoQuery[29,120,121]. GeoQuery 来源于美国的地理数据, 最初是由句子及其对应的逻辑表示构成, 之后陆续有一些工作将其逻辑形式转换为 SQL, Iyer 等人[29]和 Finegan-Dollak 等人[121]先后进行了整理和完善. 数据集包含约 880 条提问与 SQL 语句, 虽然数据规模很小, 但 SQL 语句包含了 GROUP BY、ORDER BY 和 HAVING 子句, 以及嵌套查询, 是一个单一领域且上下文无关的数据集.

3) WikiSQL[40]. ATIS 和 GeoQuery 数据集存在着数据规模小 (SQL 不足千句)、标注简单等问题. 2017 年, Zhong 等人发布了一个大型数据集 WikiSQL. WikiSQL 数据集包括 80 654 个人工标注的问题和 SQL 查询示例, 来自于 Wikipedia 的 24 241 个表. WikiSQL 的大部分问题长度为 8~15 个词, 大部分查询长度为 8~11 个词, 表中列的数量约 5~7 个. 这个大型数据集一经推出, 便引起学术界的广泛关注, 成为了最流行的 text-to-SQL 数据集. 目前, 这一数据集上最新成果的准确率已经超过了 90%. WikiSQL 的每个查询只针对一张表, 不包含排序、分组、子查询等复杂操作.

4) Spider[34]. WikiSQL 中的查询相对来说比较简单. 2018 年, 耶鲁大学的研究人员推出了 Spider 数据集. 该数据集包含了 10 181 条自然语言问句, 以及分布在 200 个独立数据库中的 5 693 条 SQL, 内容覆盖了 138 个不同的领域. 虽然在数据数量上不如 WikiSQL, 但 Spider 引入了更多的 SQL 语法, 例如 GROUP BY、ORDER BY、HAVING 等高阶操作, 甚至需要连接不同表. 相比 WikiSQL, Spider 对模型的跨领域、生成复杂 SQL 的能力提出了新的要求, 目前的最佳模型也只有 60% 左右的准确度.

5) SParC[86]. 耶鲁大学的研究人员后续又推出了 SParC, 即 Spider 的上下文相关版本. 图 13 是 SParC 数据集中的样例. 数据库基于 Spider, 但是 SParC 中的一次查询需要通过若干条相关的提问才能达成. 它包含 4000 多个连贯的问题序列, 12000 多个问题和 SQL 对.

6) CoSQL[122]. CoSQL 是用于构建跨领域会话式查询的 text-to-SQL 语料库, 它是 Spider 和 SParC 任务的会话版本. CoSQL 包含 30000 以上回合和 10000 以上带注释的 SQL 查询, 这些查询来自于 3000 个对话, 涉及 138 个领域的 200 个复杂数据库. 每个对话都模拟了一个实际的查询场景, 其中一个普通用户探索数据库, 而 SQL 专家则通过 SQL 检索答案. 除了预测本轮 SQL 外, CoSQL 还提出了两个新任务: 一是从 SQL 和查询结果生成响应, 即针对生成的 SQL 和查询结果, 产生一个自然语言描述给用户进行确认; 二是用户对话行为预测, CoSQL 为数据库用户和 SQL 专家定义了一系列对话动作.

7) 中文 CSpider[123]. 西湖大学提出了一个中文 text-to-SQL 的数据集 CSpider, 该数据集是 Spider 的中文版本. 文献 [123] 探索了中文 text-to-SQL 的一些额外的挑战, 包括中文问题到英文数据库的映射问题 (question-to-DB mapping)、中文的分词问题以及一些其他的语言现象.

目标: 在包含 5 张表的关于学生宿舍的数据库中,找到住在有电视休息室的宿舍的学生的姓氏和名字。

Q1: How many dorms have a TV lounge? (有多少间宿舍有电视休息室?)
S1: SELECT COUNT (*) FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3 ON T2.amenid=T3.amenid WHERE T3.amenity_name='TVlounge'
Q2: What is the total capacity of these dorms? (这些宿舍共能容量多少学生?)
S2:SELECT SUM (T1.student_capacity) FROM dorm AS T1 JOIN has_amentity AS T2 ON T1.dormid= T2.dormid JOIN dorm_amenity AS T3 ON T2.amenid= T3.amenid WHERE T3.amenity_name='TV Lounge'
Q3: How many students are living there? (那里住了多少学生?)
S3: SELECT COUNT (*) FROM student AS T1 JOIN lives_in AS T2 ON T1.stuid=T2.stuid WHERE T2.dormid IN (SELECT T3. dormid FROM has_amenity AS T3 JOIN dorm_amenity AS T4 ON T3.amenid=T4.amenid WHERE T4. amenity_name='TV Lounge')
Q4: Please show their first and last names.(请显示他们的名字和姓氏。)
S4: SELECT T1.fname, T1.lname FROM student AS T1 JOIN lives_in AS T2 ON T1.stuid=T2.stuid WHERE T2.dormid IN SELECT T3. dormid FROM has_amenity AS T3 JOIN dorm_amenity AS T4 ON T3.amenid=T4.amenid WHERE T4. amenity_name='TV Lounge')

图 13 SParC 数据集中的样例

8) TableQA [124]. 追一科技在 2019 年举办了首届中文 NL2SQL 挑战赛, 他们使用金融以及通用领域的表格数据作为数据源, 提供在此基础上标注的自然语言与 SQL 语句的匹配对, 包括 4 万条有标签数据作为训练集, 5000 条数据作为验证集, 1 万条无标签数据作为测试集. 与 WikiSQL 一样, 该数据集不涉及多表查询或嵌套查询.

表 2 和表 3 汇总了上述数据集的统计信息.

表 2 上下文无关的 text-to-SQL 数据集的比较

数据集	问题数量	查询数量	表格总数	数据库数量	表格数/数据库	领域	语言
GeoQuery[29,120,121]	877	246	7	1	7	单领域	英文
WikiSQL[40]	80654	77840	26531	26531	1	多领域	英文
Spider[34]	10181	5693	1020	200	5.1	多领域	英文
CSpider[123]	9691	5263	876	166	5.3	多领域	中文
TableQA [124]	64891	20311	6029	6029	1	多领域	中文

表 3 上下文相关的 text-to-SQL 数据集的比较

数据集	对话数	问题数	平均轮次	表格总数	数据库数量	表格数/数据库	领域
ATIS[118,119]	1658	11653	7.0	27	1	27	单领域
SParC[86]	4298	12726	3.0	1020	200	5.1	多领域
CoSQL[122]	3007	15598	5.2	1020	200	5.1	多领域

6.2 评价方法

text-to-SQL 任务的评价指标主要包括两种: 精确匹配率 (exact matching accuracy/logical form accuracy) 和执行准确率 (execution accuracy). 精确匹配直接比较生成的 SQL 语句与标准 SQL 语句的结构. 为了避免顺序问题造成的误判, 例如 where 子句中条件的顺序、投影列的顺序等, SQL 语句被拆解成 SELECT、WHERE、GROUP BY 等组件, 将每个组件的各成分以集合的形式表示, 并依次比较对应集合. 例如, 对于 SELECT 组件“SELECT avg(col1), max(col2), min(col1)”, 首先将其表示为集合“(avg, min, col1), (max, col2)”, 再与标准查询的集合进行比较. 仅当所有组件都与标准查询一致时, 预测的查询才是正确的. 执行准确率比较给定数据库中两个 SQL 查询的执行结果, 当预测查询与标准查询的执行结果一致时, 认为预测查询是正确的. WikiSQL 和 TableQA 数据集支持精确匹配率和执行准确率, Spider、CSpider、SParC 和 CoSQL 仅支持精确匹配率.

尽管这两种评价指标被广泛使用, 但都存在着一些问题. 精确匹配率可能将正例判负, 例如, 嵌套查询和其扁平化查询是等效的, “ORDER BY 列名 ASC LIMIT 1”和“SELECT MIN(列名)”也是等效的, 但它们的组件不同. RAT-SQL[41]在错误分析中指出, 其在 Spider 数据集上 18% 的错误查询与标准查询等效, BRIDGE[46]随机抽取了 50 个错误进行分析, 发现其中有 9 个是误判. 相同的 SQL 查询将产生相同的结果, 但两个不同的 SQL 查询偶然也会产生相同的结果, 因此执行准确率指标可能会高估系统的性能.

为了更好地评价模型的准确性, 最近一些研究工作提出了新的评价方法. Affolter 等人[125]提出了一系列复杂性不断提高的示例问题以及与之相关的数据库模式. 这种方法使其能够直接比较不同的系统, 但仅限于理论上的比较. 文献 [126] 提出了语义等价的概念, 给定两个 SQL 查询, 若它们始终在任何数据库实例上都返回相同的结果, 则它们在语义上等价. 为了考察预测 SQL 和标准 SQL 是否语义等价, 该文提出了一个多层框架, 首先比较两个 SQL 在给定数据库及生成的多个测试数据库上的执行结果, 若执行结果存在不一致的情况则一定不满足语义等价; 若生成的结果均相同, 使用 Cosette[127]进一步考察两者的句法结构是否相同, 若结构不同无法判断, 则利用查询重写器重写为标准形式, 比较它们的解析树; 若经过以上步骤仍未能确定其等价性, 则使用人工验证. 文献 [128] 从大量随机生成的数据库中提取一个小型数据库测试套件, 在该测试套件上测试 SQL 的准确性. 为了能区分错误的预测查询与标准查询, 首先生成一组邻近查询, 即每次修改标准查询中的一部分, 如数值、字符串、比较运算符、列名等, 得到形式上接近标准查询, 但在语义上不同的一组邻近查询; 之后生成大量随机数据库, 搜索并保留一小部分作为测试套件, 使得在该套件上执行标准查询与邻近查询的结果不同. 测试套件覆盖了标准查询的每个分支和子句, 可以测试每个关键子句、变量和常量的使用, 可以区分所有的邻近查询, 因此很可能也可以区分其他错误查询, 从而近似判断模型的语义准确性.

7 分析与展望

本节分析目前取得的研究进展, 总结已有的成果和存在的问题, 并对进一步的研究进行展望.

7.1 研究进展

2017 年以前, 对 text-to-SQL 任务的研究基本采用基于流水线的方法, 且集中在单领域的多表查询问题. 2017 年 WikiSQL 数据集发布, 研究热点转向基于统计学习的方法, 主要研究的问题是单轮、多领域的单表查询. 2018 年 Spider 数据集发布, 许多工作开始研究多领域、多表的单轮复杂查询问题. 到 2019 年, 开始有工作研究多轮、多领域的多表复杂查询问题. 图 14 列出了 text-to-SQL 任务主要成果的时间线.

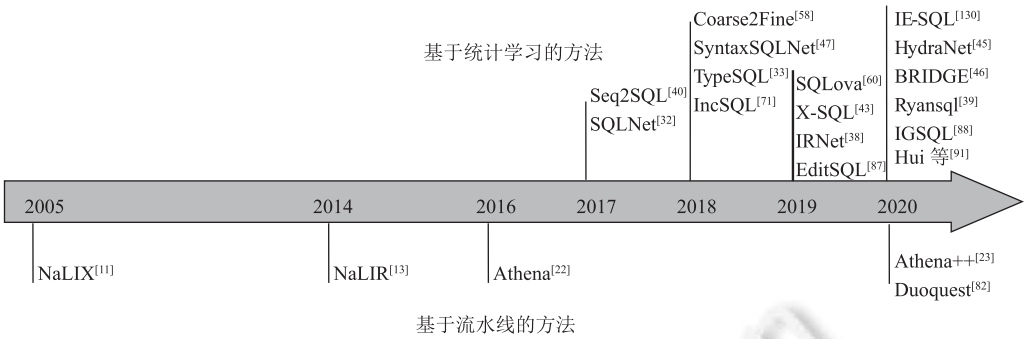


图 14 部分 text-to-SQL 模型的发展历程

对于上下文无关的单轮 text-to-SQL 任务, GeoQuery 数据集数据规模太小, WikiSQL 数据集规模最大, Spider 数据集难度较大, 更贴近真实应用场景, 因此近年来绝大部分工作均在 WikiSQL 或 Spider 数据集上进行了测试, 故下面仅对在这两个数据集上的研究进展进行总结和分析.

表 4 显示了目前在 WikiSQL 榜单[129]上排名靠前的模型, 这些模型都是基于统计学习的方法. 目前排名第一的解决方案是 IE-SQL[130]+Execution-Guided Decoding[70], 其在测试集上的精确匹配率和执行准确率分别为 87.8% 和 92.5%.

表 4 WikiSQL 验证集和测试集上不同模型的精确匹配度和执行准确率 (%)

模型	验证集		测试集	
	精确匹配度	执行准确率	精确匹配度	执行准确率
IE-SQL+BERT+EG[130]	87.9	92.6	87.8	92.5
HydraNet+RoBERTa+EG [45]	86.6	92.4	86.5	92.2
BRIDGE+EG[46]	86.8	92.6	86.3	91.9
X-SQL+MT-DNN+EG[43]	86.2	92.3	86.0	91.8
SQLova+BERT+EG[60]	84.2	90.2	83.6	89.6
IncSQL+EG [71]	51.3	87.2	51.1	87.1
Coarse2Fine[58]	72.5	79.0	71.7	78.5
TypeSQL[33]	—	74.5	—	73.5
SQLNet[32]	63.2	69.8	61.3	68.0

WikiSQL 仅涉及单表上的简单查询, 查询仅包含 SELECT 和 WHERE 子句, 不含 ORDER BY、GROUP BY HAVING、嵌套查询, 任务难度低, 表 4 中除 IncSQL[71]、BRIDGE[46]、IE-SQL[130]外, 均采用基于子任务的解码方法. SQLNet[32]将 SQL 生成分解为 6 个子任务, 除了 WHERE 子句中的值由指针网络预测外, 其他子任务使用专用分类器进行预测, 取得了不错的效果, TypeSQL[33]、SQLova[60]、X-SQL[43]和 HydraNet[45]均继承了 SQLNet[32]对任务划分的思路. 后续工作对 SQLNet[32]的改进主要集中两点: 增加数据库和 NLQ 的对齐, 以及建模解码子任务间的依赖性. TypeSQL[33]针对 NLQ 中稀有实体和数字识别, 利用数据库内容和知识库识别 NLQ 中的实体并标记其类型. Coarse2Fine[58]没有将 Where 子句的生成拆分为多个任务, 其首先生成粗略的中间输出, 再细化结果. IncSQL[71]使用序列到动作的解码模型, 保证了解码的上下文依赖. SQLova[60]使用了预训练模型加强 NLQ 和数据库的对齐, SQLNet[32]推断 WHERE 子句中值仅依赖于 WHERE 子句中的列, SQLova[60]根据列和操作符推断条件值在 NLQ 中的起始位置和结束位置. X-SQL[43]在预训练模型基础上, 还增加了上下文增强层增强模式表示, 并提出一种基于 KL 散度的目标函数解决子任务分别优化的问题. SQLova[60]和 X-SQL[43]是将 NLQ 和所有列串联送入预训练模型编码的, 而 HydraNet[45]编码时一次只处理一个列, 这与训练 BERT/RoBERTa 的原始任务完全吻合, 避免了池化层或其他编码层, 解码时首先计算每个“NLQ-列”对的所有子任务结果, 对于 WHERE 条件数量和

SELECT 数量, 对所有列结果加权得到. 基于子任务的解码方式可以有效简化问题, 减小解码搜索解空间, 但没有考虑子任务之间的依赖关系. 针对这一问题, IE-SQL[130]没有使用编码器-解码器框架, 而是使用了一种新颖的基于序列标注的提取链接 (extraction-linking) 方法, 由提取器采用序列标注的方法识别 NLQ 中提到的槽以及它们的关系, 并基于关联规则更新聚集函数的预测. 由于 NLQ 可能不会显式提及列名, 或字符串形式与数据库模式中实际列名不同, 需要使用链接器将列的提及与模式进行映射, 得到可执行的 SQL 查询. 由于在问题中, 值、操作对应的触发词和对应列经常一起出现, 因此序列标注模型可以更好地捕捉它们的依赖关系, 有利于对它们的识别.

表 5 显示的是 Spider 榜单[131]上排名靠前的模型, 目前排名第一的解决方案是 RAT-SQL[41] + GraPPa[79], 其在测试集上的精确匹配率为 69.6%. 与表 4 类似, Spider 榜单上基于统计学习的方法也明显占优, 除 ATHENA++[23]在 Spider 验证集上的准确率达 78.89% 外, 根据文献 [126] 的结果, 在其他数据集上基于流水线的方法也缺乏竞争力.

表 5 Spider 验证集和测试集上不同模型的精确匹配度 (%)

模型	验证集	测试集
RAT-SQL+ GraPPa[79]	73.43	69.63
RAT-SQL+BERT[41]	69.7	65.6
BRIDGE+BERT[46]	71.1	67.5
Ryansql+BERT [39]	66.6	58.2
SLSQL+BERT[132]	60.8	55.7
IRNet+BERT[38]	61.9	54.7
EditSQL+BERT[87]	57.6	53.4
Global-GNN[68]	52.1	47.4
GNN[37]	40.7	39.4
RCSQL[36]	28.8	24.3
SyntaxSQLNet[47]	24.8	27.2
SQLNet[32,34,131]	10.9	12.4
TypeSQL[33,34,131]	8.2	8.0

与 WikiSQL 数据集相比, Spider 数据集上的成绩总体上较差. 这是因为 WikiSQL 数据集仅涉及含聚集、比较和选择操作的单表查询, 而 Spider 数据集涉及多表的复杂查询. 一方面问题与数据库的对齐变得困难, 无法明确问题涉及的模式项自然难以生成正确的查询; 另一方面, 复杂查询的生成对模型又提出了较高的要求. 后续工作针对 Spider 任务的特点, 围绕编码和解码方法进行了改进, 使得结果有了显著的提升.

(1) 在编码方面, 一些先进的模型使用了预训练模型和统一编码的方法. 预训练语言模型 (如 Transformer) 可以增强对自然语言变化的概括, 捕获句子内长期依赖关系, 可以显著提高模型的准确性. 统一编码的方式可以增强 NLQ 与数据库之间的对齐或映射关系, 帮助模型理解问题中涉及了哪些表、哪些列, 问题中的哪些词语触发了查询的关键操作, 如聚集、分组、排序等, 结合预训练模型能取得更好的效果.

文献 [37] 使用 GNN 对数据库模式的编码进行了改进, 数据库模式能够反映数据的结构及其联系, 因此数据库模式也会对 SQL 语句产生影响. 即使问题的句式相似, 但是由于数据库模式不同, 作为结果的 SQL 语句也有很大的差距, GNN[37]使用图神经网络可以更好地捕捉数据库的主外键关系. IRNet[38]首次提出了模式链接模块的概念, 使用字符串匹配和知识图谱进行数据库和 NLQ 的对齐. SLSQL[132]使用预训练模型和注意力机制进行模式链接, 并进行了对比实验, 证明了精确的模式链接可以提高 SQL 解析的性能. Ryansql[39]使用字符嵌入和 CNN, 提高 NLQ 和模式项部分匹配的识别能力, 同时加入了两个对齐层加强列和表的编码. RAT-SQL[41]和 BRIDGE[46]都对 NLQ 和数据库模式进行统一编码, 并且都使用了数据库的内容加强对齐. 在数据隐私允许的情况下, 使用数据库字段内容可以提升模型的识别能力, 有时问题中不会显式提及表名或列名, 数据库字段内容可以更好地帮助模型理解问题.

(2) 在解码方面, 最近的很多模型针对复杂 SQL 引入了更多的子任务, 并特别通过语法约束解码搜索空间. 如

Ryansql[39]的子任务多达 11 个, 任务间的依赖关系相应的更加复杂. BRIDGE[46]、 EditSQL[87]、SLSQL[132]使用了改进的 seq-to-seq 方法进行解码, 在解码过程中, 会应用语法的约束来修剪预测空间, 例如列会在 SELECT 或逗号之后生成. 这种方式可以捕捉任务间的依赖关系, 缺点是不好处理顺序问题. EditSQL[87]的解码器根据问题和上下文生成列名或 SQL 关键字, SLSQL[132]在其基础上采用了一个两阶段策略, 先生成没有聚合函数的 SQL 序列, 再生成聚合函数, BRIDGE[46]根据 SQL 执行的特点, 对子句生成的顺序进行了重排. SyntaxSQLNet[47]首先使用了 seq-to-tree 的解码思路, 后续许多工作如 GNN[37]、RAT-SQL[41]、IRNet[38]等在 Spider 任务上使用 seq-to-tree 的解码方法, 通过语法约束解码搜索空间, SQL 语言本身具有层次结构, seq-to-tree 中层次结构的解码器使得模型学习到了 SQL 的结构, 这些结构信息结合注意力机制学习输入到 SQL 语义表示的软对齐, 取得了不错的效果.

在其他方面, 表 5 中 IRNet[38]不是端到端模型, 其首先使用神经网络生成一种中间的语义表示形式 SemQL, 然后再将中间表示转换成 SQL 查询语句, 可以视为结合流水线的混合模型, SemQL 采用树状结构, 使其很容易转换成 SQL 语句. 另外许多工作都使用了执行指导[70]技术, 在解码过程中检测并排除错误结果.

表 6 列出了目前在 WikiSQL 和 Spider 数据集上表现较好的模型及其主要贡献.

表 6 不同模型的技术比较和主要贡献

模型	编码		解码	主要贡献
	骨干网络	NLQ和数据库编码		
RAT-SQL[41]	Transformer, Bi-LSTM	统一编码	seq-to-tree	统一框架进行模式链接和表结构编码, 基于Transformer, 增加关系感知自注意力机制
BRIDGE[46]	Transformer, Bi-LSTM	统一编码	改进的seq-to-seq	联合表示NLQ、数据库模式和值的通用框架, 提取锚文本形成混合标记序列, 加强NLQ和数据库对齐
Ryansql[39]	NLQ: CNN, 模式: CNN+Transformer	分别编码+注意力	基于子任务	针对复杂查询设计子任务, 设计语句位置编码, 通过递归来解决子查询的问题
SLSQL[132]	Transformer, Bi-GRU	统一编码	改进的seq-to-seq	在BERT基础上增加模式链接模块, 研究了模式链接的作用
IRNet[38]	Bi-LSTM	分别编码+字符串匹配	seq-to-tree	提出模式链接, 提出NLQ和SQL的中间表示SemQL
EditSQL[87]	Bi-LSTM	分别编码+注意力	改进的seq-to-seq	提出查询编辑机制, 在生成本轮的SQL语句时, 同时重用上一轮的SQL语句
Global-GNN[68]	NLQ: Bi-LSTM, 模式: GNN	分别编码+注意力	seq-to-tree	在GNN[37]基础上, 全局选择与问题相关的列/表, 对候选结果进行重排
GNN[37]	NLQ: Bi-LSTM, 模式: GNN	分别编码+注意力	seq-to-tree	数据库结构用图神经网络建模
RCSQL[36]	Bi-LSTM	分别编码	基于子任务	根据子句任务创建解码器, 使用递归方法生成子查询
SyntaxSQLNet [47]	Bi-LSTM	分别编码	seq-to-tree	第1个针对Spider任务设计的模型, 基于树结构生成SQL, 用数据增强来提升模型表现
IESQL[130]	—	—	—	提出提取-链接方法, 基于序列标注的方法提取问题中提到的列名、值、操作符和聚合函数
HydraNet[45]	Transformer	统一编码	基于子任务	每次编码NLQ和一列, 将解码任务分为与具体列有关和无关的两类
X-SQL[43]	Transformer	统一编码	基于子任务	利用预训练模型和上下文增强层增强模式表示, 针对子任务分别优化提出基于列表的全局排名方法
SQLova[60]	Transformer	统一编码	基于子任务	证明了学习高度上下文相关和表格感知的单词表示的重要性, 探索了基于 BERT编码器的3种解码器变体
IncSQL[71]	Bi-LSTM	分别编码+注意力	seq-to-action	提出使用非确定性预言机训练序列到动作模型
Coarse2Fine[58]	Bi-LSTM	分别编码+注意力	基于子任务	提出了一种从粗到精的语义解析解码框架.
TypeSQL[33]	Bi-LSTM	分别编码	基于子任务	从知识图谱或数据库内容中提取类型信息, 使得模型更好地理解NLQ中稀有实体和数字
SQLNet[32]	Bi-LSTM	分别编码	基于子任务	采用槽填充的思想, 首次将SQL生成解耦为6个子任务, 避免了WHERE语句的顺序问题

上下文相关的多轮 text-to-SQL 任务结合了对话理解和 text-to-SQL 自身的难点, 难度较大, 相关研究也不是很多, 目前在 SParC 和 CoSQL 数据集上的准确率大约为 40% 左右. 表 7 列出了目前最优模型的精确匹配度结果, 其中问题精确匹配度 (question match) 是针对所有问题的精确匹配度分数, 交互精确匹配度 (interaction match) 以一个轮次为单位, 需要在一轮对话中的每个问题都匹配成功才得分. EditSQL[87]考虑了用户的历史输入和先前预测的查询, IGSQL[88]在其基础上加入数据库模式项的历史信息, 使用前一轮和当前轮中相邻的模式项更新当前轮的模式项表示, Hui 等人[91]将上下文话语、词例、数据库模式及其交互信息统一到一个动态图框架中, 是目前最先进的解决方案. CoSQL 提出了两个新任务, 从 SQL 和查询结果生成响应, 以及用户对话行为预测, 目前还没有与之相关的研究工作.

表 7 最前沿的模型在 SParC 和 CoSQL 测试集上的表现 (%)

模型	SParC		CoSQL	
	问题精确匹配度	交互精确匹配度	问题精确匹配度	交互精确匹配度
Hui等人[91]	55.8	30.8	46.8	17.0
IGSQL[88]	51.2	29.5	42.5	15.0
EditSQL[87]	47.9	25.3	40.8	13.7
GAZP[76]	45.9	23.5	39.7	12.8

7.2 存在的问题

虽然目前对 text-to-SQL 的研究取得了较大的进步, 但仍有提升的空间, 目前存在的问题可以总结如下.

(1) 自然语言问题与数据库模式的对齐映射

识别自然语言问题中的实体, 并将其映射到数据库中的表或列, 是 text-to-SQL 任务中一个非常关键且困难的问题. 如前文所述, IRNet[38]将该问题命名为模式链接, 目前各研究工作提出的解决方案包括: 使用预训练模型、问题和数据库的联合编码、对问题和模式项进行字符串匹配以及注意力机制. 即便有上述这些解决方案, 模式链接问题仍然是当前限制模型效果的主要问题之一, RAT-SQL[41]指出其错误中 39% 涉及 SELECT 子句中的列, 包括生成了错误的列或无关的列, 以及列的缺失, 是由于模式链接机制的限制导致的. Ryansql[39]也指出在其分析的错误中, 列选择错误占 34.9%. 模式链接的困难主要包括以下几个方面: 1) 自然语言表述具有多样性, 对于同一实体或概念存在不同形式的表达, 不能期望用户准确的提及列名或表名, 例如“公司”又可以表述为“厂商”“供应商”, “负债率”“资产负债率”“举债经营比率”是同一概念. 文献 [133] 将 Spider 数据集中的问题进行了同义词替换, IRNet[38]、GNN[37]和 RAT-SQL[41]的精确匹配率均大幅下降. 2) 问题中的单个实体可能在数据库中对应多个模式项, 例如问题: “各种职称教师的平均工资是多少? ”, 其中“工资”对应到数据库中的“基本工资”和“绩效工资”两列. 3) 有时问题中不会显式提及表名或列名, 例如问题: “联想今年出货量是多少? ”, 并未提及联想为公司名称, 又如“2019 年房地产公司的平均主营业务收入是多少? ”, 其中“房地产”对应“公司”表中的“行业名称”列. 4) 数据库中可能存在相似的表名和列名, 使对齐时有歧义, 例如在 Spider 数据集的“car_1”数据库中, 表“car_makers”和“model_list”中都含有名为“Maker”的列, “car_names”中又包含列“MakeId”和“Make”.

此外, Spider 中平均每个数据库含 5 张表, 平均每个 SQL 对应 1.8 个问题, 而现实世界中的数据库表格数更多, 模式更复杂, 典型的企业数据库中列名称相似性高, 包含很多稀有词或缩写, 在单个表中具有 30-40 列和数百万行, 同一查询存在着多种提问的表述方式, 这将使得问题与数据库的匹配对齐更加困难.

(2) 跨领域语言理解

当模型在训练时未见过的数据库上进行测试时, 领域外单词、短语的理解是一大难点. 例如问题: “给出所有运动员的平均年龄”, 年龄需要用当前年份减去“运动员”表“出生日期”列中的年份值, 如果模型在训练时未见过询问“年龄”的样本, 那么很可能无法计算出正确的值. 又如问题“校体育场可以坐多少人? ”, 其中“坐多少人”需要对“体育场”表的“容量”列进行查询. 此外, 相同的词语在不同领域表达的意思可能不同, 例如训练集中询问“最大的

企业”, 其中“最大”指的是企业的市值, 而测试集中询问“最大的岛”, 模型可能无法理解此处“最大”指的是面积. RAT-SQL[41]中 Where 条件错误占 29%, 而特定领域的用语理解是造成此类错误的常见原因, 如模型无法理解“older than 21”中的 older 一词, 无法生成正确的条件“age > 21”. BRIDGE[46]也提到训练时未见过的词汇理解问题是 SQL 生成错误的主要原因之一, 例如问题“What are the full names of all left handed players, in order of birth date?”, 模型未理解“left handed”, 造成结果缺少了条件“WHERE hand = ‘L’”.

(3) 复杂查询生成

目前的模型还不能很好地解决涉及复杂查询的问题, 如含有聚合函数的多表联接、包含 EXCEPT 或嵌套子查询的 SQL 语句等. Spider 数据集根据 SQL 组件的难度及数量, 将问题难度分为简单、中等、困难、特别困难 4 个等级, RAT-SQL+BERT[41]在验证集上由易到难 4 个级别的准确率分别为 86.4%、73.6%、62.1% 和 42.9%, BRIDGE[46]的结果则分别为 85.5%、71.5%、56.3% 和 51.8%, 由此可见复杂查询的生成仍有提高的空间.

对于表连接, 外键数量越多、表越多的数据库, 查询时越有可能需要连接更多的表, 模型必须从许多候选中选择正确的表连接路径, 对模型捕获表之间关系的能力提出了巨大挑战. 以 CSpider 中的样本为例, 问题“找出‘1970’年生产汽车的制造商的名字”, 对应的 SQL 为:

```
SELECT DISTINCT T1.Maker
FROM CAR_MAKERS AS T1 JOIN MODEL_LIST AS T2 ON T1.Id = T2.Maker JOIN CAR_NAMES AS T3 ON T2.model = T3.model JOIN CARS_DATA AS T4 ON T3.MakeId = T4.id
WHERE T4.year = ‘1970’;
```

这个查询需要对 4 张表进行连接操作. 而问题“列出广州恒大队在季后赛获胜时的最高得分”, 需要对没有主外键关系的“postseason”和“team”两表进行连接, 这无疑更加困难, SLSQL[132]不支持这样的查询, 问题“显示高中学生小军的所有朋友的名字”, 由于高中生的朋友也是高中生, 需要对表“Highschooler”进行自连接, 更具挑战性.

对于包含子查询的问题, 如何识别输入问题与子查询的对应关系, 明确每个子查询的组成部件, 这些都存在一定的难度. CSpider 中的问题“找出没有记录的游泳运动员的名字”, 对应的查询为:

```
SELECT name FROM swimmer WHERE id NOT IN (SELECT swimmer_id FROM record)
```

而对于问题“找出没有现金支付的客户的名字”, 其对应 SQL 为:

```
SELECT customer_name FROM customers WHERE payment_method != ‘现金’
```

虽然两个问题的句式类似, 但实现的方式截然不同, 模型需要检测何时需要构造子查询, 且使用问题中并未提及的“swimmer_id”构造子查询.

由此可见, 使用自然语言进行提问时, 不会包含查询实现的具体方法, 构建复杂查询需要模型具有一定的推理能力, 自然语言中的歧义使得理解查询的语义是一个巨大的挑战.

(4) 条件值的生成

条件值的生成困难主要在于问题中的描述可能与数据库中存储的值不一致, 包括缩写、单位不一致、别名等. 例如问题: “鹅厂的平均工资是多少？”, 其中“鹅厂”是“腾讯公司”的别名, 但是并未出现在数据库中. 目前很少有工作针对该问题进行研究, 大多都采用指针机制, 原因主要有两个: 第一, Spider 数据集不支持执行准确率, 且在精确匹配率计算时仅考察 SQL 结构, 不考虑 SQL 中的值, 因此许多模型忽略了该问题. 第二, 根据 TableQA[124]的统计, WikiSQL 和 Spider 数据集中超过 97% 和 95% 的问题在提及条件值时, 其表述与 SQL 语句中的表述完全相同, 该问题并没有显现出来.

7.3 研究展望

text-to-SQL 的研究工作仍面临诸多挑战, 下一步可以从以下方面来展开研究.

(1) 向多轮对话扩展

用户在描述问题时, 可能会存在语法错误、拼写错误或使用带有歧义性、含混不清的表达, 然而目前许多基于深度学习的方法, 因为可解释性差, 很难向用户解释并获得反馈对结果进行改进. 若查询的问题很复杂, 用户很

难用一句话加以描述, 因此多轮对话更符合实际场景需求, 这要求模型能够处理连续的上下文, 虽然已经有一些在 SParC 和 CoSQL 数据集上的研究, 但准确率都很低. 此外, 系统如何指导用户提出可以回答的问题, 优化与用户交互的次数也值得研究. 捕获领域语义并将其融合到对话中并非易事, 缺乏足够的训练数据也是挑战之一.

(2) 引入外部知识

许多查询的生成需要特定领域的知识或常识, 例如自然语言问题中的“年龄”对应数据库中的列“出生年月”, 问题中的“1 万”在数据库中表示为“10 000”, 查询当前债务余额时, 要根据提款还本金额进行计算等, 这些都需要引入外部知识, 用于捕获领域语义, 理解领域词汇及其同义词. 当然, 如何将外部知识引入到模型中来是一个挑战.

(3) 混合方法

基于流水线的方法比较适合有较为明确步骤的算法, 而基于统计学习的方法适合学习比较抽象的知识, 因此, 可以对混合方法进行更多研究, 以充分利用各自方法的优势. 例如, 自然连接中连接条件的生成可以不用统计学习的方法, 因为只要正确地预测出了表名, 并理解了数据库模式, 那么就可以根据模式来自动生成连接条件.

(4) 与基于文本的问答结合

从用户的角度, 用户并不关心生成的 SQL 是什么, 而只关心最终的查询结果是什么. 从这个意义上来说, text-to-SQL 类似于问答, 只不过这个答案需要在数据库里查找. 因此, 可以试图与基于文本的问答技术相结合. 现实中, 结构化数据和文本是需要同时考虑的. 例如有些金融数据既有结构化数据也有文本, 需要将两者结合才能得出完整答案. 而有些数据库中包含了很多文本, 也需要结合文本问答才能得出细粒度的答案.

总的来说, text-to-SQL 的研究在过去几年的进展非常大, 进一步的提升可以借助于语言的理解、知识的运用, 以及多种任务和技术的综合使用, 而这些问题的研究涉及很多其他领域. 随着研究人员对于语言、知识的理解不断加深, 以及相关技术不断成熟, text-to-SQL 有望取得突破, 并得到大规模的应用.

References:

[1] Androutsopoulos I, Ritchie GD, Thanisch P. Natural language interfaces to databases—An introduction. Natural Language Engineering, 1995, 1(1): 29–81. [doi: 10.1017/S135132490000005X]

[2] Warren DHD, Pereira FCN. An efficient easily adaptable system for interpreting natural language queries. Computational Linguistics, 1982, 8(3–4): 110–122.

[3] Popescu AM, Armanasu A, Etzioni O, Ko D, Yates A. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In: Proc. of the 20th Int'l Conf. on Computational Linguistics. Geneva: ACL, 2004. 141. [doi: 10.3115/1220355.1220376]

[4] Popescu AM, Etzioni O, Kautz H. Towards a theory of natural language interfaces to databases. In: Proc. of the 8th Int'l Conf. on Intelligent User Interfaces. New York: ACM, 2003. 149–157. [doi: 10.1145/604045.604070]

[5] Giordani A, Moschitti A. Translating questions to SQL queries with generative parsers discriminatively reranked. In: Proc. of the COLING 2012: Posters. Mumbai: The COLING 2012 Organizing Committee, 2012. 401–410.

[6] Baik C, Jagadish HV, Li YY. Bridging the semantic gap with SQL query logs in natural language interfaces to databases. In: Proc. of the 35th IEEE Int'l Conf. on Data Engineering (ICDE). Macao: IEEE, 2019. 374–385. [doi: 10.1109/ICDE.2019.00041]

[7] Simitsis A, Koutrika G, Ioannidis Y. Précis: From unstructured keywords as queries to structured databases as answers. The VLDB Journal, 2008, 17(1): 117–149. [doi: 10.1007/s00778-007-0075-9]

[8] Blunschi L, Jossen C, Kossmann D, Mori M, Stockinger K. SODA: Generating SQL for business users. Proc. of the VLDB Endowment, 2012, 5(10): 932–943. [doi: 10.14778/2336664.2336667]

[9] Tata S, Lohman GM. SQAK: Doing more with keywords. In: Proc. of the 2008 ACM SIGMOD Int'l Conf. on Management of Data. New York: Association for Computing Machinery, 2008. 889–902. [doi: 10.1145/1376616.1376705]

[10] Kaufmann E, Bernstein A, Fischer L. NLP-reduce: A "naive" but domain-independent natural language interface for querying ontologies. In: Proc. of the 4th European Semantic Web Conf. Innsbruck: Springer, 2007. 1–2.

[11] Li YY, Yang HH, Jagadish HV. NaLIX: An interactive natural language interface for querying XML. In: Proc. of the 2005 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM, 2005. 900–902. [doi: 10.1145/1066157.1066281]

[12] Li F, Jagadish HV. Constructing an interactive natural language interface for relational databases. Proc. of the VLDB Endowment, 2014, 8(1): 73–84. [doi: 10.14778/2735461.2735468]

[13] Li F, Jagadish HV. Understanding natural language queries over relational databases. ACM SIGMOD Record, 2016, 45(1): 6–13. [doi: 10.1145/2949741.2949744]

[14] Waltinger U, Tecuci D, Olteanu M, Mocanu V, Sullivan S. USI answers: Natural language question answering over (semi-) structured industry data. In: Proc. of the 27th AAAI Conf. on Artificial Intelligence. Bellevue: AAAI Press, 2013. 1471–1478.

[15] Song DZ, Schilder F, Smiley C, Brew C, Zielund T, Bretz H, Martin R, Dale C, Duprey J, Miller T, Harrison J. TR discover: A natural language interface for querying and analyzing interlinked datasets. In: Proc. of the 14th Int'l Semantic Web Conf. Bethlehem: Springer, 2015: 21–37. [doi: 10.1007/978-3-319-25010-6_2]

[16] Singh G, Solanki A. An algorithm to transform natural language into SQL queries for relational databases. Selforganizology, 2016, 3(3): 100–116.

[17] Jamil HM. Knowledge rich natural language queries over structured biological databases. In: Proc. of the 8th ACM Int'l Conf. on Bioinformatics, Computational Biology, and Health Informatics. New York: ACM, 2017. 352–361. [doi: 10.1145/3107411.3107442]

[18] Yaghmazadeh N, Wang YP, Dillig I, Dillig T. SQLizer: Query synthesis from natural language. Proc. of the ACM on Programming Languages, 2017, 1: 63. [doi: 10.1145/3133887]

[19] Zheng WG, Cheng H, Zou L, Yu JX, Zhao KF. Natural language question/answering: Let users talk with the knowledge graph. In: Proc. of the 2017 ACM on Conf. on Information and Knowledge Management. New York: Association for Computing Machinery, 2017. 217–226. [doi: 10.1145/3132847.3132977]

[20] Deutch D, Frost N, Gilad A. Provenance for natural language queries. Proc. of the VLDB Endowment, 2017, 10(5): 577–588. [doi: 10.14778/3055540.3055550]

[21] Sukthankar N, Maharnawar S, Deshmukh P, Haribhakta Y, Kamble V. NQuery—A natural language statement to SQL query generator. In: Proc. of the 2017 ACL, Student Research Workshop. Vancouver: Association for Computational Linguistics, 2017. 17–23. [doi: 10.18653/v1/P17-3004]

[22] Saha D, Floratou A, Sankaranarayanan K, Minhas UF, Mittal AR, Özcan F. ATHENA: An ontology-driven system for natural language querying over relational data stores. Proc. of the VLDB Endowment, 2016, 9(12): 1209–1220. [doi: 10.14778/2994509.2994536]

[23] Sen J, Lei C, Quamar A, Özcan F, Efthymiou V, Dalmia A, Stager G, Mittal A, Saha D, Sankaranarayanan K. ATHENA++: Natural language querying for complex nested SQL queries. Proc. of the VLDB Endowment, 2020, 13(12): 2747–2759. [doi: 10.14778/3407790.3407858]

[24] Sen J, Babtiwale T, Saxena K, Butala Y, Bhatia S, Sankaranarayanan K. Schema aware semantic reasoning for interpreting natural language queries in enterprise settings. In: Proc. of the 28th Int'l Conf. on Computational Linguistics. Barcelona: Int'l Committee on Computational Linguistics, 2020. 1334–1345. [doi: 10.18653/v1/2020.coling-main.115]

[25] Lin JF, Liu YL, Guo J, Cleland-Huang J, Goss W, Liu WC, Lohar S, Monaikul N, Rasin A. Tiqi: A natural language interface for querying software project data. In: Proc. of the 32nd IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). Urbana: IEEE, 2017. 973–977. [doi: 10.1109/ASE.2017.8115714]

[26] Wang XY, Xu JQ, Lu H. NALMO: A natural language interface for moving objects databases. In: Proc. of the 17th Int'l Symp. on Spatial and Temporal Databases. New York: ACM, 2021. 1–11. [doi: 10.1145/3469830.3470894]

[27] Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, Hellmann S, Morsey M, Van Kleef P, Auer S, Bizer C. DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web, 2015, 6(2): 167–195. [doi: 10.3233/sw-140134]

[28] Miller GA. WordNet: A lexical database for English. Communications of the ACM, 1995, 38(11): 39–41. [doi: 10.1145/219717.219748]

[29] Iyer S, Konstas I, Cheung A, Krishnamurthy J, Zettlemoyer L. Learning a neural semantic parser from user feedback. In: Proc. of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vancouver: Association for Computational Linguistics, 2017. [doi: 10.18653/v1/P17-1089]

[30] Basik F, Hättasch B, Ilkhechi A, Usta A, Ramaswamy S, Utama P, Weir N, Binnig C, Cetintemel U. DBPal: A learned NL-interface for databases. In: Proc. of the 2018 Int'l Conf. on Management of Data. New York: ACM, 2018: 1765–1768. [doi: 10.1145/3183713.3193562]

[31] Dong Z, Sun SZ, Liu HZ, Lou JG, Zhang DM. Data-anonymous encoding for text-to-SQL generation. In: Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int'l Joint Conf. on Natural Language Processing (EMNLP-IJCNLP). Hong Kong: ACL, 2019: 5404–5413. [doi: 10.18653/v1/D19-1543]

[32] Xu XJ, Liu C, Song D. SQLNet: Generating structured queries from natural language without reinforcement learning. arXiv:1711.04436, 2018.

[33] Yu T, Li ZF, Zhang ZL, Zhang R, Radev D. TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In: Proc. of the

2018 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). New Orleans: Association for Computational Linguistics, 2018. 588–594. [doi: 10.18653/v1/N18-2093]

[34] Yu T, Zhang R, Yang K, Yasunaga M, Wang DX, Li ZF, Ma J, Li I, Yao QN, Roman S, Zhang ZL, Radev D. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In: Proc. of the 2018 Conf. on Empirical Methods in Natural Language Processing. Brussels: Association for Computational Linguistics, 2018. 3911–3921. [doi: 10.18653/v1/D18-1425]

[35] Lin ZH, Feng MW, Dos Santos CN, Yu M, Xiang B, Zhou BW, Bengio Y. A structured self-attentive sentence embedding. arXiv:1703.03130, 2017.

[36] Lee D. Clause-wise and recursive decoding for complex and cross-domain text-to-SQL generation. In: Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int'l Joint Conf. on Natural Language Processing (EMNLP-IJCNLP). Hong Kong: Association for Computational Linguistics, 2019. 6045–6051. [doi: 10.18653/v1/D19-1624]

[37] Bogin B, Berant J, Gardner M. Representing schema structure with graph neural networks for text-to-SQL parsing. In: Proc. of the 57th Annual Meeting of the Association for Computational Linguistics. Florence: ACL, 2019. 4560–4565. [doi: 10.18653/v1/P19-1448]

[38] Guo JQ, Zhan ZC, Gao Y, Xiao Y, Lou JG, Liu T, Zhang DM. Towards complex text-to-SQL in cross-domain database with intermediate representation. In: Proc. of the 57th Annual Meeting of the Association for Computational Linguistics. Florence: ACL, 2019. 4524–4535. [doi: 10.18653/v1/p19-1444]

[39] Choi D, Shin MC, Kim E, Shin DR. RYANSQL: Recursively applying sketch-based slot fillings for complex text-to-SQL in cross-domain databases. arXiv:2004.03125, 2020.

[40] Zhong V, Xiong CM, Socher R. Seq2SQL: Generating structured queries from natural language using reinforcement learning. arXiv:1709.00103, 2017.

[41] Wang BL, Shin R, Liu XD, Polozov O, Richardson M. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2020. 7567–7578. [doi: 10.18653/v1/2020.acl-main.677]

[42] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I. Attention is all you need. In: Proc. of the 31st Int'l Conf. on Neural Information Processing Systems. Red Hook: Curran Associates Inc., 2017. 6000–6010.

[43] He PC, Mao Y, Chakrabarti K, Chen WZ. X-SQL: Reinforce schema representation with context. arXiv:1908.08113v1, 2019.

[44] Liu XD, He PC, Chen WZ, Gao JF. Multi-task deep neural networks for natural language understanding. In: Proc. of the 57th Annual Meeting of the Association for Computational Linguistics. Florence: Association for Computational Linguistics, 2019. 4487–4496. [doi: 10.18653/v1/P19-1441]

[45] Lyu Q, Chakrabarti K, Hathi S, Kundu S, Zhang JW, Chen Z. Hybrid ranking network for text-to-SQL. arXiv:2008.04759, 2020.

[46] Lin XV, Socher R, Xiong CM. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In: Findings of the Association for Computational Linguistics: EMNLP 2020. Association for Computational Linguistics, 2020. 4870–4888. [doi: 10.18653/v1/2020.findings-emnlp.438]

[47] Yu T, Yasunaga M, Yang K, Zhang R, Wang DX, Li ZF, Radev D. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In: Proc. of the 2018 Conf. on Empirical Methods in Natural Language Processing. Brussels: Association for Computational Linguistics, 2018. 1653–1663. [doi: 10.18653/v1/D18-1193]

[48] Cai RC, Xu BY, Zhang ZJ, Yang XY, Li ZJ, Liang ZH. An encoder-decoder framework translating natural language to database queries. In: Proc. of the 27th Int'l Joint Conf. on Artificial Intelligence. Stockholm: IJCAI, 2018: 3977–3983. [doi: 10.24963/ijcai.2018/553]

[49] Liu HY, Fang L, Liu Q, Chen B, Lou JG, Li ZJ. Leveraging adjective-noun phrasing knowledge for comparison relation prediction in text-to-SQL. In: Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int'l Joint Conf. on Natural Language Processing (EMNLP-IJCNLP). Hong Kong: ACL, 2019. 3515–3520. [doi: 10.18653/v1/D19-1356]

[50] Tian Y, Shou LD, Chen K, Luo XY, Chen G. Natural language interface for databases with content-based table column embeddings. Computer Science, 2020, 47(9): 60–66 (in Chinese with English abstract). [doi: 10.11896/jsjkx.190800138]

[51] Wang WL, Tian YT, Wang HX, Ku WS. A natural language interface for database: Achieving transfer-learnability using adversarial method for question understanding. In: Proc. of the 36th IEEE Int'l Conf. on Data Engineering (ICDE). Dallas: IEEE, 2020. 97–108. [doi: 10.1109/icde48307.2020.00016]

[52] Vinyals O, Fortunato M, Jaitly N. Pointer networks. arXiv:1506.03134, 2017.

[53] Gulcehre C, Ahn S, Nallapati R, Zhou BW, Bengio Y. Pointing the unknown words. In: Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin: ACL, 2016. 140–149. [doi: 10.18653/v1/P16-1014]

[54] Wang CL, Brockschmidt M, Singh R. Pointing out SQL queries from text. 2017. https://www.microsoft.com/en-us/research/publication/

pointing-sql-queries-text/

[55] Lukovnikov D, Chakraborty N, Lehmann J, Fischer A. Translating natural language to SQL using pointer-generator networks and how decoding order matters. arXiv:1811.05303, 2018.

[56] Sun YB, Tang DY, Duan N, Ji JS, Cao GH, Feng XC, Qin B, Liu T, Zhou M. Semantic parsing with syntax- and table-aware SQL generation. In: Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne: ACL, 2018. 361–372. [doi: 10.18653/v1/P18-1034]

[57] Wang P, Shi T, Reddy CK. A translate-edit model for natural language question to SQL query generation on multi-relational healthcare data. arXiv:1908.01839, 2019.

[58] Dong L, Lapata M. Coarse-to-fine decoding for neural semantic parsing. In: Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne: ACL, 2018: 731–742. [doi: 10.18653/v1/P18-1068]

[59] Guo T, Gao HL. Bidirectional attention for SQL generation. arXiv:1801.00076, 2018.

[60] Hwang W, Yim J, Park S, Seo M. A comprehensive exploration on wikisql with table-aware word contextualization. arXiv:1902.01069, 2019.

[61] Zhang XY, Yin FJ, Ma GJ, Ge B, Xiao WD. M-SQL: Multi-task representation learning for single-table text2sql generation. IEEE Access, 2020, 8: 43156–43167. [doi: 10.1109/access.2020.2977613]

[62] Zhang XY, Yin FJ, Ma GJ, Ge B, Xiao WD. F-SQL: Fuse table schema and table content for single-table text2sql generation. IEEE Access, 2020, 8: 136409–136420. [doi: 10.1109/access.2020.3011747]

[63] Cao JC, Huang T, Chen G, Wu XF, Chen K. Research on Technology of Generating multi-table SQL query statement by natural language. Journal of Frontiers of Computer Science & Technology, 2020, 14(7): 1133–1141 (in Chinese with English abstract). [doi: 10.3778/j.issn.1673-9418.1908025]

[64] Li Q, Zhong J, Li LL, Li Q, Zhang SF, Zhang J. Semantic parsing graph query model for on-demand aggregation. Acta Electronica Sinica, 2019, 48(4): 763–771 (in Chinese with English abstract). [doi: 10.3969/j.issn.0372-2112.2020.04.018]

[65] Lin K, Bogin B, Neumann M, Berant J, Gardner M. Grammar-based neural text-to-SQL generation. arXiv:1905.13326, 2019.

[66] Liang C, Norouzi M, Berant J, Le Q, Lao N. Memory augmented policy optimization for program synthesis and semantic parsing. In: Proc. of the 32nd Int'l Conf. on Neural Information Processing Systems. Montréal, 2018. 10015–10027.

[67] Agarwal R, Liang C, Schuurmans D, Norouzi M. Learning to generalize from sparse and underspecified rewards. In: Proc. of the 36th Int'l Conf. on Machine Learning. Long Beach: PMLR, 2019. 130–140.

[68] Bogin B, Gardner M, Berant J. Global reasoning over database structures for text-to-SQL parsing. In: Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int'l Joint Conf. on Natural Language Processing (EMNLP-IJCNLP). Hong Kong: ACL, 2019: 3659–3664. [doi: 10.18653/v1/D19-1378]

[69] Kelkar A, Relan R, Bhardwaj V, Vaichal S, Khatri C, Relan P. Bertrand-DR: Improving text-to-SQL using a discriminative re-ranker. arXiv:2002.00557, 2020.

[70] Wang CL, Tatwawadi K, Brockschmidt M, Huang PS, Mao Y, Polozov O, Singh R. Robust text-to-SQL generation with execution-guided decoding. arXiv:1807.03100, 2018.

[71] Shi TZ, Tatwawadi K, Chakrabarti K, Mao Y, Polozov O, Chen WZ. IncSQL: Training incremental text-to-SQL parsers with non-deterministic oracles. arXiv:1809.05054, 2018.

[72] Huang PS, Wang C, Singh R, Yih WT, He XD. Natural language to structured query generation via meta-learning. In: Proc. of the 2018 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). New Orleans: ACL, 2018. 732–738. [doi: 10.18653/v1/N18-2115]

[73] Lee D, Yoon J, Song JY, Lee S, Yoon S. One-shot learning for text-to-SQL generation. arXiv:1905.11499, 2019.

[74] Chang SC, Liu FP, Tang Y, Huang J, He XD, Zhou BW. Zero-shot text-to-SQL learning with auxiliary task. arXiv:1908.11052, 2019.

[75] Guo DY, Sun Y, Tang DY, Duan N, Yin J, Chi H, Cao J, Chen P, Zhou M. Question generation from SQL queries improves neural semantic parsing. In: Proc. of the 2018 Conf. on Empirical Methods in Natural Language Processing. Brussels: Association for Computational Linguistics, 2018. 1597–1607. [doi: 10.18653/v1/D18-1188]

[76] Zhong V, Lewis M, Wang SI, Zettlemoyer L. Grounded adaptation for zero-shot executable semantic parsing. In: Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2020. 6869–6882. [doi: 10.18653/v1/2020.emnlp-main.558]

[77] Yin PC, Neubig G, Yih W, Riedel S. TaBERT: Pretraining for joint understanding of textual and tabular data. In: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2020. 8413–8426. [doi: 10.18653/v1/2020.acl-main.745]

[78] Herzig J, Nowak PK, Müller T, Piccinno F, Eisenschlos J. TAPAS: Weakly supervised table parsing via pre-training. In: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2020. 4320–4333. [doi: 10.18653/v1/2020.acl-main.398]

[79] Yu T, Wu CS, Lin XV, Wang BL, Tan YC, Yang XY, Radev D, Socher R, Xiong CM. GraPPa: Grammar-augmented pre-training for table semantic parsing. arXiv:2009.13845, 2020.

[80] McCann B, Keskar NS, Xiong CM, Socher R. The natural language decathlon: Multitask learning as question answering. arXiv:1806.08730, 2018.

[81] Gur I, Yavuz S, Su Y, Yan XF. DialSQL: Dialogue based structured query generation. In: Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne: Association for Computational Linguistics, 2018. 1339–1349. [doi: 10.18653/v1/P18-1124]

[82] Baik C, Jin ZJ, Cafarella M, Jagadish HV. Duoquest: A dual-specification system for expressive SQL queries. In: Proc. of the 2020 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM, 2020: 2319–2329. [doi: 10.1145/3318464.3389776]

[83] Li YT, Chen B, Liu Q, Gao Y, Lou JG, Zhang Y, Zhang DM. "What do you mean by that?" a parser-independent interactive approach for enhancing text-to-SQL. In: Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2020. 6913–6922. [doi: 10.18653/v1/2020.emnlp-main.561]

[84] Elgohary A, Hosseini S, Awadallah AH. Speak to your parser: Interactive text-to-SQL with natural language feedback. In: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 2020. 2065–2077. [doi: 10.18653/v1/2020.acl-main.187]

[85] Suhr A, Iyer S, Artzi Y. Learning to map context-dependent sentences to executable formal queries. In: Proc. of the 2018 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). New Orleans: Association for Computational Linguistics, 2018. 2238–2249. [doi: 10.18653/v1/N18-1203]

[86] Yu T, Zhang R, Yasunaga M, Tan YC, Lin XV, Li SY, Er HY, Li I, Pang B, Chen T, Ji E, Dixit S, Proctor D, Shim S, Kraft J, Zhang V, Xiong CM, Socher R, Radev D. SParC: Cross-domain semantic parsing in context. In: Proc. of the 57th Annual Meeting of the Association for Computational Linguistics. Florence: ACL, 2019. 4511–4523. [doi: 10.18653/v1/p19-1443]

[87] Zhang R, Yu T, Er HY, Shim S, Xue E, Lin XV, Shi TZ, Xiong CM, Socher R, Radev D. Editing-based SQL query generation for cross-domain context-dependent questions. In: Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int'l Joint Conf. on Natural Language Processing (EMNLP-IJCNLP). Hong Kong: Association for Computational Linguistics, 2019. 5338–5349. [doi: 10.18653/v1/D19-1537]

[88] Cai YT, Wan XJ. IGSQL: Database schema interaction graph based neural model for context-dependent text-to-SQL generation. In: Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2020. 6903–6912. [doi: 10.18653/v1/2020.emnlp-main.560]

[89] Wang RZ, Ling ZH, Zhou JB, Hu Y. Tracking interaction states for multi-turn text-to-SQL semantic parsing. In: Proc. of the 35th AAAI Conf. on Artificial Intelligence. Palo Alto: AAAI Press, 2020. 13979–13987.

[90] Liu Q, Chen B, Guo JQ, Lou JG, Zhou B, Zhang DM. How far are we from effective context modeling? An exploratory study on semantic parsing in context. In: Proc. of the 29th Int'l Joint Conf. on Artificial Intelligence. Yokohama: IJCAI, 2020. 3580–3586.

[91] Hui BY, Geng RY, Ren QY, Li BH, Li YB, Sun J, Huang F, Si L, Zhu PF, Zhu XD. Dynamic hybrid relation network for cross-domain context-dependent semantic parsing. arXiv:2101.01686, 2021.

[92] Wong YW, Mooney R. Learning synchronous grammars for semantic parsing with lambda calculus. In: Proc. of the 45th Annual Meeting of the Association of Computational Linguistics. Prague: Association for Computational Linguistics, 2007. 960–967.

[93] Yih SW, Chang MW, He XD, Gao JF. Semantic parsing via staged query graph generation: Question answering with knowledge base. In: Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int'l Joint Conf. on Natural Language Processing (Volume 1: Long Papers). Beijing: Association for Computational Linguistics, 2015. 1321–1331. [doi: 10.3115/v1/p15-1128]

[94] Matuszek C, Herbst E, Zettlemoyer L, Fox D. Learning to Parse Natural Language Commands to a Robot Control System. In: Desai JP, Dudek G, Khatib O, Kumar V, eds. Experimental Robotics. Heidelberg: Springer, 2013: 403–415. [doi: 10.1007/978-3-319-00065-7_28]

[95] Yih WT, Richardson M, Meek C, Chang MW, Suh J. The value of semantic parse labeling for knowledge base question answering. In: Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Berlin: Association for Computational Linguistics, 2016. 201–206. [doi: 10.18653/v1/p16-2033]

[96] Matuszek C, FitzGerald N, Zettlemoyer L, Bo LF, Fox D. A joint model of language and perception for grounded attribute learning. arXiv:1206.6423, 2012.

[97] Misra DK, Tao KJ, Liang P, Saxena A. Environment-driven lexicon induction for high-level instructions. In: Proc. of the 53rd Annual

Meeting of the Association for Computational Linguistics and the 7th Int'l Joint Conf. on Natural Language Processing (Volume 1: Long Papers). Beijing: Association for Computational Linguistics, 2015. 992–1002. [doi: 10.3115/v1/p15-1096]

[98] Woods W. The lunar sciences natural language information system. BBN report, Bolt Beranek and Newman, 1972. https://ci.nii.ac.jp/naid/10010193391/

[99] Zettlemoyer LS, Collins M. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In: Proc. of the 21st Conf. on Uncertainty in Artificial Intelligence. Arlington: AUAI Press, 2005. 658–666.

[100] Liang PS. Learning dependency-based compositional semantics [Ph.D. Thesis]. Berkeley: University of California, 2011.

[101] Zettlemoyer L, Collins M. Online learning of relaxed CCG grammars for parsing to logical form. In: Proc. of the 2007 Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL). Prague: Association for Computational Linguistics, 2007. 678–687.

[102] Kwiatkowski T, Zettlemoyer L, Goldwater S, Steedman M. Lexical generalization in CCG grammar induction for semantic parsing. In: Proc. of the 2011 Conf. on Empirical Methods in Natural Language Processing. Edinburgh: Association for Computational Linguistics, 2011. 1512–1523.

[103] Jia RB, Liang P. Data recombination for neural semantic parsing. In: Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin: Association for Computational Linguistics, 2016. 12–22. [doi: 10.18653/v1/P16-1002]

[104] Dong L, Lapata M. Language to logical form with neural attention. In: Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin: Association for Computational Linguistics, 2016. 33–43. [doi: 10.18653/v1/P16-1004]

[105] Xiao CY, Dymetman M, Gardent C. Sequence-based structured prediction for semantic parsing. In: Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Berlin: Association for Computational Linguistics, 2016. 1341–1350. [doi: 10.18653/v1/P16-1127]

[106] Rabinovich M, Stern M, Klein D. Abstract syntax networks for code generation and semantic parsing. In: Proc. of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vancouver: Association for Computational Linguistics, 2017. 1139–1149. [doi: 10.18653/v1/P17-1105]

[107] Yin PC, Neubig G. A syntactic neural model for general-purpose code generation. In: Proc. of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vancouver: Association for Computational Linguistics, 2017. 440–450. [doi: 10.18653/v1/P17-1041]

[108] Krishnamurthy J, Dasigi P, Gardner M. Neural semantic parsing with type constraints for semi-structured tables. In: Proc. of the 2017 Conf. on Empirical Methods in Natural Language Processing. Copenhagen: Association for Computational Linguistics, 2017. 1516–1526. [doi: 10.18653/v1/D17-1160]

[109] Chen B, Sun L, Han XP. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. In: Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne: Association for Computational Linguistics, 2018. 766–777. [doi: 10.18653/v1/P18-1071]

[110] Artzi Y, Zettlemoyer L. Weakly supervised learning of semantic parsers for mapping instructions to actions. Trans. of the Association for Computational Linguistics, 2013, 1: 49–62. [doi: 10.1162/tacl_a_00209]

[111] Reddy S, Lapata M, Steedman M. Large-scale semantic parsing without question-answer pairs. Trans. of the Association for Computational Linguistics, 2014, 2: 377–392. [doi: 10.1162/tacl_a_00190]

[112] Berant J, Chou A, Frostig R, Liang P. Semantic parsing on freebase from question-answer pairs. In: Proc. of the 2013 Conf. on Empirical Methods in Natural Language Processing. Seattle: Association for Computational Linguistics, 2013. 1533–1544.

[113] Berant J, Liang P. Semantic parsing via paraphrasing. In: Proc. of the 2nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Baltimore: Association for Computational Linguistics, 2014. 1415–1425. [doi: 10.3115/v1/P14-1133]

[114] Herzig J, Berant J. Decoupling structure and lexicon for zero-shot semantic parsing. In: Proc. of the 2018 Conf. on Empirical Methods in Natural Language Processing. Brussels: Association for Computational Linguistics, 2018. 1619–1629. [doi: 10.18653/v1/D18-1190]

[115] Sun YB, Tang DY, Duan N, Gong YY, Feng XC, Qin B, Jiang DX. Neural semantic parsing in low-resource settings with back-translation and meta-learning. Proc. of the AAAI Conf. on Artificial Intelligence, 2020, 34(5): 8960–8967. [doi: 10.1609/aaai.v34i05.6427]

[116] Ni A, Yin P, Neubig G. Merging weak and active supervision for semantic parsing. Proc. of the AAAI Conf. on Artificial Intelligence, 2020, 34(5): 8536–8543. [doi: 10.1609/aaai.v34i05.6375]

[117] Shao B, Gong YY, Qi WZ, Duan N, Lin XL. Multi-level alignment pretraining for multi-lingual semantic parsing. In: Proc. of the 28th Int'l Conf. on Computational Linguistics. Barcelona: Int'l Committee on Computational Linguistics, 2020. 3246–3256. [doi: 10.18653/v1/2020.coling-main.289]

[118] Hemphill CT, Godfrey JJ, Doddington GR. The ATIS spoken language systems pilot corpus. In: Proc. of a Workshop on Speech and Natural Language. Pennsylvania: Association for Computational Linguistics, 1990. 96–101. [doi: 10.3115/116580.116613]

[119] Dahl DA, Bates M, Brown MK, Fisher WM, Hunicke-Smith K, Pallett DS, Pao C, Rudnicky A, Shriberg E. Expanding the scope of the ATIS task: The atis-3 corpus. In: Proc. of the Workshop on Human Language Technology. New Jerey: Association for Computational Linguistics, 1994. 43–48. [doi: 10.3115/1075812.1075823]

[120] Zelle JM, Mooney RJ. Learning to parse database queries using inductive logic programming. In: Proc. of the 13th National Conf. on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conf. Portland: AAAI Press, 1996. 1050–1055.

[121] Finegan-Dollak C, Kummerfeld JK, Zhang L, Ramanathan K, Sadasivam S, Zhang R, Radev D. Improving text-to-SQL evaluation methodology. In: Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Melbourne: ACL, 2018. 351–360. [doi: 10.18653/v1/P18-1033]

[122] Yu T, Zhang R, Er HY, Li SY, Xue E, Pang B, Lin XV, Tan YC, Shi TZ, Li ZH, Jiang YX, Yasunaga M, Shim S, Chen T, Fabbri A, Li ZF, Chen LY, Zhang YW, Dixit S, Zhang V, Xiong CM, Socher R, Lasecki WS, Radev D. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In: Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int'l Joint Conf. on Natural Language Processing (EMNLP-IJCNLP). Hong Kong: Association for Computational Linguistics, 2019. 1962–1979. [doi: 10.18653/v1/D19-1204]

[123] Min QK, Shi YF, Zhang Y. A pilot study for Chinese SQL semantic parsing. In: Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int'l Joint Conf. on Natural Language Processing (EMNLP-IJCNLP). Hong Kong: Association for Computational Linguistics, 2019. 3652–3658. [doi: 10.18653/v1/D19-1377]

[124] Sun NY, Yang XF, Liu YF. Tableqa: A large-scale Chinese text-to-SQL dataset for table-aware SQL generation. arXiv:2006.06434, 2020.

[125] Affolter K, Stockinger K, Bernstein A. A comparative survey of recent natural language interfaces for databases. The VLDB Journal, Springer, 2019, 28(5): 793–819. [doi: 10.1007/s00778-019-00567-8]

[126] Kim H, So BH, Han WS, Lee H. Natural language to SQL: Where are we today? Proc. of the VLDB Endowment, 2020, 13(10): 1737–1750. [doi: 10.14778/3401960.3401970]

[127] Chu SM, Wang CL, Weitz K, Cheung A. Cosette: An automated prover for SQL. In: Proc. of the 8th Biennial Conf. on Innovative Data Systems Research. Chaminade: CIDR, 2017.

[128] Zhong RQ, Yu T, Klein D. Semantic evaluation for text-to-SQL with distilled test suites. In: Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2020. 396–411. [doi: 10.18653/v1/2020.emnlp-main.29]

[129] WikiSQL. https://github.com/salesforce/WikiSQL

[130] Ma JQ, Yan Y, Pang S, Zhang Y, Shen JP. Mention extraction and linking for SQL query generation. In: Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2020. 6936–6942. [doi: 10.18653/v1/2020.emnlp-main.563]

[131] Spider 1.0. https://yale-lily.github.io/spider

[132] Lei WQ, Wang WX, Ma ZX, Gan T, Lu W, Kan MY, Chua TS. Re-examining the role of schema linking in text-to-SQL. In: Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2020. 6943–6954. [doi: 10.18653/v1/2020.emnlp-main.564]

[133] Gan YJ, Chen XY, Huang QP, Purver M, Woodward JR, Xie JX, Huang PS. Towards robustness of text-to-SQL models against synonym substitution. In: Proc. of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th Int'l Joint Conf. on Natural Language Processing (Volume 1: Long Papers). Association for Computational Linguistics, 2021. 2505–2515. [doi: 10.18653/v1/2021.acl-long.195]

附中文参考文献:

[50] 田野, 寿黎但, 陈珂, 骆歆远, 陈刚. 基于字段嵌入的数据库自然语言查询接口. 计算机科学, 2020, 47(9): 60–66. [doi: 10.11896/jsjkx.190800138]

[63] 曹金超, 黄滔, 陈刚, 吴晓凡, 陈珂. 自然语言生成多表SQL查询语句技术研究. 计算机科学与探索, 2020, 14(7): 1133–1141. [doi: 10.

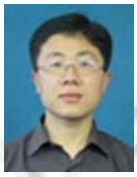
3778/j.issn.1673-9418.1908025]
[64] 李青, 钟将, 李立力, 李琪, 张淑芳, 张剑. 一种依需聚合的语义解析图查询模型. 电子学报, 2019, 48(4): 763–771. [doi: 10.3969/j.issn.0372-2112.2020.04.018]

刘喜平(1981－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为信息检索, 数据库, 数据挖掘.

舒晴(1990－), 女, 博士生, CCF 学生会员, 主要研究领域为机器学习, 数据挖掘.

何佳壕(1994－), 男, 硕士生, CCF 学生会员, 主要研究领域为文本挖掘, 数据挖掘.

万常选(1962－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为 Web 数据管理, 情感分析, 数据挖掘, 信息检索.



刘德喜(1975－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为社会媒体处理, 信息检索, 自然语言处理.