中国科学: 信息科学 2022 年 第 52 卷 第 8 期: 1363–1392
SCIENTIA SINICA Informationis
《中国科学》杂志社
SCIENCE CHINA PRESS
评述

基于深度学习的 SQL 生成研究综述

梁清源[1], 朱琪豪[2], 孙泽宇[2], 张路[2*], 张文杰[2], 熊英飞[2], 梁广泰[3], 郁莲[1]

1. 北京大学软件与微电子学院, 北京 102600
2. 高可信软件技术教育部重点实验室 (北京大学), 北京 100871
3. 华为云软件分析实验室, 北京 100095
* 通信作者. E-mail: zhanglu@sei.pku.edu.cn

收稿日期: 2021–09–13; 修回日期: 2021–11–10; 接受日期: 2022–01–19; 网络出版日期: 2022–08–03

国家重点研发计划 (批准号: 2019YFE0198100)、香港特别行政区创新科技署 (批准号: MHP/055/19) 和国家自然科学基金 (批准号: 61872011) 资助项目

摘要 SQL 生成 (text-to-SQL) 是自动化软件工程的重要应用之一, 也是语义解析领域的研究热点. SQL 生成根据输入的自然语言描述自动生成相应的 SQL 数据库查询语句, 它允许非专业人员在不了解 SQL 语法的情况下访问数据库. 随着大量 SQL 相关数据集的不断构造以及人工智能技术的卓越进步, SQL 生成任务也得到了极大的发展. 基于深度学习的 SQL 生成 (deep learning-based text-to-SQL) 能够利用大规模数据的优势, 从已有数据中学习自然语言、数据库以及 SQL 语句的表示, 并根据新的自然语言输入生成符合查询需求的 SQL 语句. 相对于传统的 SQL 生成, 基于深度学习的 SQL 生成具有高准确率、输入信息灵活和可迭代学习的优点. 近年来, 研究者在基于深度学习的 SQL 生成方面进行了一系列的研究, 本文从 SQL 生成场景、数据集、模型结构和评估方法层面对现有研究进行分类综述.

关键词 SQL 生成, 语义解析, 深度学习, 代码生成, 编码 – 解码模型

1 引言

SQL (structured query language) 是操作关系型数据库的结构化查询语言, 用于访问和处理结构化数据. 与命令式编程语言不同, SQL 的本质是一种基于集合的声明式编程语言. SQL 在 1986 年成为美国国家标准化组织 (American National Standards Institute, ANSI) 的一项标准, 并在 1987 年成为国际标准化组织 (International Organization for Standardization, ISO) 的一项标准 [1], 后续发展中标准 SQL 也存在多种扩展形式. SQL 相比较于之前的读写接口而言, 可以允许一条命令访问多个记录, 并且消除了指定如何到达记录的必要性, 例如, 可以不使用索引. 在现代软件中, 当涉及关系型数据库

引用格式: 梁清源, 朱琪豪, 孙泽宇, 等. 基于深度学习的 SQL 生成研究综述. 中国科学: 信息科学, 2022, 52: 1363–1392, doi: 10.1360/SSI-2021-0316
Liang Q Y, Zhu Q H, Sun Z Y, et al. A survey of deep learning based text-to-SQL generation (in Chinese). Sci Sin Inform, 2022, 52: 1363–1392, doi: 10.1360/SSI-2021-0316

的操作时, 往往都会用到 SQL 语句, 如操作电商平台存储用户和商品数据、查询学校中老师和课程信息等.

然而, 对于不了解 SQL 语法的人, 很难在包含大量记录的数据库中快速地找到自己感兴趣的数据. 在具体应用场景中, 重复性地数据查询往往也会消耗大量的人力资源, 如客户向电商客服询问具体商品价格和参数. 因此, 通过对所需要查询的意图进行描述, 由 SQL 生成 (text-to-SQL) 模型自动生成具有相应功能的 SQL 语句是必要且非常有帮助的, 它能够让非专业人员轻松地访问数据库, 降低大规模数据分析的门槛, 也可以在减少人力消耗的情况下提供自动且及时的查询数据反馈. 除此之外, SQL 语法和模式相对于命令式编程语言而言较为简单, 生成 SQL 语句的任务也有较高的可行性.

SQL 生成任务旨在根据已有数据库信息, 将自然语言描述映射到特定功能的 SQL 语句, 它是软件工程领域中代码生成任务以及自然语言处理领域中语义解析 (semantic parsing) 任务的热点研究之一. SQL 生成任务允许使用查询问题的语义来访问和操作结构化数据库信息, 而无需了解复杂的 SQL 查询语法, 极大地减少了查询数据库所需要的先验知识. 已有研究包含了使用基于规则 [2]、程序合成 [3] 等方法生成查询语句, 但这些方法通常只在小规模数据集上实验并且难以达到较好的生成效果. 随着深度学习技术的不断涌现以及研究人员对 SQL 生成场景的不断探索, 基于深度学习的 SQL 生成 (deep learning-based text-to-SQL) 成为了当前非常重要的研究问题. 近年来基于深度学习的 SQL 生成研究取得了大量的研究成果, 生成 SQL 的效果不断提升, 成为了 SQL 生成研究中主流的方式. 因此, 针对基于深度学习的 SQL 生成问题进行综述对代码自动化生成和深度学习应用具有重要的研究意义.

目前也有研究针对 SQL 生成问题进行综述 [4, 5]. 但主要存在如下问题: (1) 现有分类框架较为粗糙, 无法全面覆盖 SQL 生成研究; (2) 对特定类别下的研究文献整理不够充分, 没有涉及目前最新的方法和模型; (3) 没有整理现有研究的具体思路. 本综述基于这些问题, 构造了系统的分类框架对现有研究进行整理, 并且分析了现有 SQL 生成研究的常见思路. 本文将基于深度学习的 SQL 生成的研究分为 4 个部分, 分别是 SQL 生成场景、数据集、模型结构和评估指标. 其中, SQL 生成场景包含了常见 SQL 生成中所涉及的需求和要素; 数据集可以根据所涉及的数据库是否包含不同领域信息分为单领域数据集和跨域数据集; 模型结构可以分为编码模型和解码模型, 编码模型可以根据所处理对象是否结构化分为自然语言描述编码、数据库编码和自然语言与数据库之间的模式链接编码 3 部分, 数据库编码又可以细分为数据模式和数据库记录的编码, 解码模型可以根据 SQL 不同的生成方式分为模版插值的方式和基于生成的方式, 基于生成的方式包括了基于序列生成和基于语法生成两种具体的方法.

本文对基于深度学习的 SQL 生成研究工作进行综述. 首先整理了 SQL 生成常见场景要素. 接着探索了基于深度学习的 SQL 生成方法的数据集, 根据不同数据集的特点归纳了数据集的研究和发展. 然后本文对深度学习模型结构进行了分析, 从编码模型和解码模型两方面入手对现有方法建立了详细的分类系统, 并对每一个详细类别的研究工作进行综述. 另外, 本文也整理了针对 SQL 生成模型的评估指标. 最后讨论了基于深度学习的 SQL 生成研究面临的主要挑战和未来研究方向.

2 基于深度学习的 SQL 生成分类方法和相关工作

2.1 基本概念

为了便于理解, 本文将 SQL 生成相关概念做如下介绍.

(1) SQL 生成 (text-to-SQL): 本综述针对从文本转化为 SQL 语句的任务, 该任务旨在将用户输入

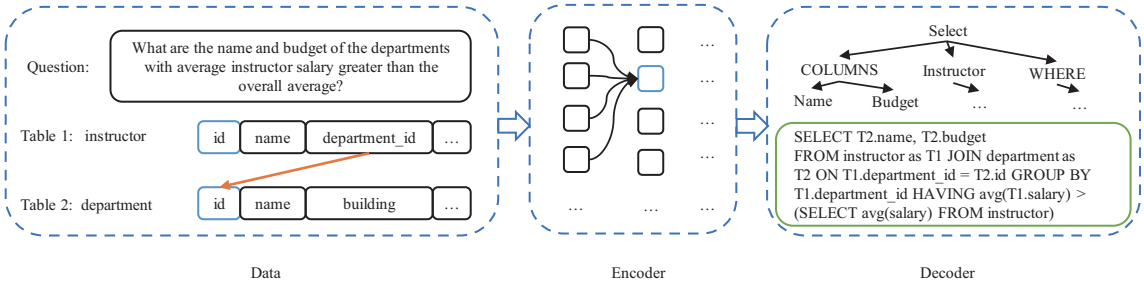

图 1 (网络版彩图) 基于深度学习的方法生成 SQL 语句的示例

Figure 1 (Color online) An example of generating a SQL statement based on deep learning

的查询问题转化为相应的 SQL 查询语句.

(2) 语义解析 (semantic parsing): 将自然语言形式的输入映射到可执行的程序, 其中可执行程序的形式包括逻辑表达式、SQL 查询语句、Python 程序、JAVA 程序等.

(3) 编码模型 (encoding model): 将数据输入给深度模型, 并使用深度学习的编码器 (encoder) 编码为模型内部的表示形式.

(4) 解码模型 (decoding model): 利用深度学习的解码器 (decoder) 将模型内部表示形式解码为目标形式, 如 SQL 语句、Python 代码等.

(5) 抽象语法树 (abstract syntax tree, AST): 抽象语法树是源代码语法结构的一种抽象表示. 它以树状的结构表示编程语言的语法结构, 树上的每个节点都表示源代码中的一种结构.

(6) SQL 语法: SQL 语言的语法通常由抽象语法描述语言 [6] (abstract syntax description language, ASDL) 定义. 在给定语法的情况下, 一个 SQL 语句可以解析为相应的抽象语法树. 研究者也可以自定义 SQL 语法.

(7) 模版插值 (sketch-based slot-filling): 模版插值是生成 SQL 的一种方法, 这类方法的特点是先创建一个通用的 SQL 模版, 并将需要生成的部分空出来, 之后训练模型对空出来的位置进行填空, 最后组成一个完整的 SQL 语句

2.2 研究概况

图 1 展示了一个基于深度学习的方法生成 SQL 语句的示例, SQL 语句生成过程主要涉及数据集和模型结构中的编码、解码模型, 关于 SQL 生成研究的创新目前也集中在以上 3 个方向. 数据集为深度学习模型提供用于训练的语料, 其中数据类型和特点能够反映 SQL 的具体生成场景. 不同数据集决定了模型不同的应用场景和应用范围, 并且对数据集的研究也极大地促进了后续 SQL 生成方法的创新. 目前基于深度学习的方法基本遵循编码器 – 解码器结构, 涉及编码模型和解码模型两个阶段, 后续研究通常针对这两个阶段进行优化.

可以从 SQL 生成场景中数据库领域对现有数据集进行划分. 数据集的早期研究专注在各个不同领域内部收集查询问题和相应 SQL 语句, 致力于提升单领域数据集质量. 在近期的研究中, 大多数研究者更加关注大规模的跨领域数据集, 通过提升数据复杂度增加 SQL 生成任务的挑战性, 以此来激发研究者构建更好的深度学习模型. 另外, 也有研究者在大规模跨域数据集的基础上增加查询上下文依赖的情况, 构造了交互式生成 SQL 语句的数据集. 总之, 随着越来越多数据集的提出, 不仅增加了研究者对 SQL 生成任务的研究兴趣, 也增加对深度学习模型生成能力的挑战性.

编码模型的研究主要关注如何对查询问题以及 SQL 生成任务中特有的结构化数据库进行合适的

编码. 具体而言, 编码过程需要同时考虑查询问题、数据库以及查询问题和数据库之间的模式链接, 其中模式编码和模式链接对生成 SQL 语句非常重要, 也是编码模型的关键研究问题. 因此, 图网络、预训练模型等先进的深度学习技术也不断被引入到 SQL 生成任务, 以此提升模型编码能力. 为了使得编码过程能够更好地覆盖查询问题和数据库之间的联系, 也有越来越多的研究开始使用丰富模型的编码内容, 以及使用专门的预训练模型增强处理自然语言与数据库模式之间链接的能力.

解码模型的研究也是 SQL 生成任务的重点之一, 它决定了如何将编码后的信息转化为 SQL 语句或其等价表示形式. 编码器 – 解码器结构的基本输出方式为顺序地生成普通单词序列, 但模型解码后常常无法识别 SQL 语句内不需要顺序限制的部分, 如 SQL 语句中并列的查询条件, 以及生成不符合语法的 SQL 语句. 基于这些问题, 研究者在解码模型部分提出了一系列基于模版的方法以及使用含语法约束或结构敏感的解码器等研究, 以此提升模型解码能力, 并尽可能保证模型生成有意义的可执行程序.

整体而言, SQL 生成的研究是当前语义解析任务的前沿研究. 相比较于生成其他命令式编程语言的程序, SQL 生成任务的效果相对显著, 能够在高准确率的情况下生成较为复杂的 SQL 语句. 同时 SQL 生成也是面向实际应用方面最具有潜力的语义解析任务.

2.3 工业界经典落地项目

在大数据时代, 对感兴趣数据进行快速抽取和数据分析逐渐成为人们关注的焦点. 但对数据库的分析和操作通常需要具有编程语言基础 (如 SQL), 这给非专业人员设置了较高的门槛. 目前工业界提出了一系列基于 SQL 生成技术的对话式数据分析应用, 只需输入自然语言, 就可以获取和分析大规模的数据. 下面将从典型落地场景中的智能客服和 Excel 中智能数据分析举例.

在智能客服的应用中, SQL 生成与智能对话系统的结合技术能够 24 小时自动查询数据库并回答用户相关问题, 极大地减少了人工客服的消耗和运营成本, 能够有效地提升用户体验. 阿里巴巴的小蜜智能客服是其中的典型代表, 它能够帮助用户解决常用的业务咨询类问题, 以及通过多轮对话完成商品导购的工作. 尤其是在大型购物节期间, 如双十一购物节, 这种智能客服能够及时响应, 有效减少了因为客服人数不足而导致的阻塞和商业损失. 在这种智能问答情况下, 用户一般看不到后台数据库, 查找相应商品信息又很耗时, 基于自然语言理解的智能模型能够有效提升用户体验, 为企业创造更多商业价值.

Excel 作为通用数据处理和分析软件, 已经成为了人们日常办公中最重要的软件. 但很多用户通常只使用 Excel 的基础功能, 对数据分析方面仍存在一定的难度. 微软亚洲研究院开发了一套智能分析算法 AnnaParser 为 Excel 提供对话式智能分析功能. AnnaParser 是一种典型的 SQL 生成的落地算法, 即通过自然语言的理解生成 SQL 处理数据并输出处理后结果. AnnaParser 首先使用数据抽象模块 (data abstraction) 来识别自然语言与数据模式和数据记录的联系, 并替换成相应符号输入语义解析模块. 然后使用知识理解 (knowledge understanding) 模块挖掘每个表格背后隐含的知识, 最后使用一种自底向上的解析框架来生成最终的 SQL 语句进行数据的处理和分析. 类似地, 工业界内的谷歌、Tableau、SalesForce 等公司同样也一直关注对话式数据分析并活跃在学术前沿.

总而言之, SQL 生成研究具有现实意义并能够在工业界中的大型项目中落地, 能够实际提升用户在特定需求下的应用体验, 并且可以提升企业相应商品的竞争力和商业价值.

2.4 分类方法

本文对基于深度学习的 SQL 生成研究所涉及的 SQL 生成场景、数据集、模型结构和评估方法进

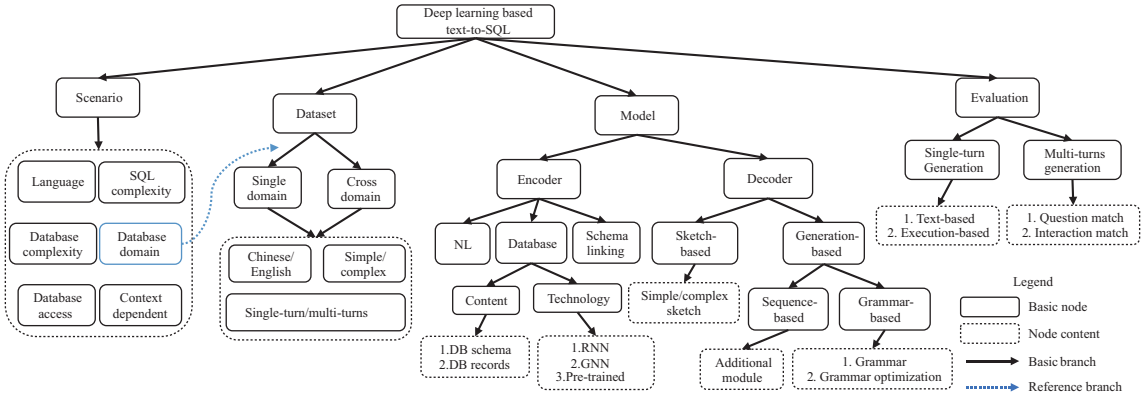

图 2 (网络版彩图) 基于深度学习的 SQL 生成系统分类框架

Figure 2 (Color online) Taxonomy of deep learning-based SQL generation systems

行了详细的划分和总结. 最终得到相应的研究分类框架如图 2 所示.

场景方面, SQL 生成场景反映了生成过程中面临的具体需求和特点, 场景可以根据不同场景元素加以区分, 如场景中的语种、是否有多轮 SQL 生成等. 一个数据集可以包含实际场景中的多种元素, 根据 SQL 生成不同的需求和特点, 可以选择和组合不同的场景元素, 从而形成符合特定需求的 SQL 生成任务. 之后的数据集研究将根据 SQL 生成场景元素中数据是否包含多个领域的数据库进行划分. 在划分后依然可以从其他场景元素的角度继续分析, 如数据涉及的语种以及是否需要使用多轮 SQL 生成的方式.

数据集方面, SQL 生成数据集可以从是否包含不同领域数据库的角度划分, 分为单领域数据集和跨域数据集. 在单领域的数据集中, 模型只需要针对单个领域数据库的查询问题生成相应的 SQL 语句. 单领域的数据集只涉及单一数据库以及单一的领域知识, 因此所需数据数量相对较少, 模型学习相对简单, 同时应用范围也有限. 在跨域的数据集中, 模型不仅要学习多领域数据库所涉及的知识, 并要求模型能够泛化到训练过程未见过的数据库. 由于涉及的数据库和领域知识更多, 跨域数据集的数量也相对较多, 对模型的学习和泛化能力要求更高, 但同时跨域数据集的应用价值相比单领域数据集更大. 跨域数据集内部也可以根据场景元素的语种、SQL 语句复杂程度以及生成方式是否涉及多轮生成继续进行划分和组合.

模型结构方面, 可根据编码模型和解码模型两部分展开. 编码模型可以从编码的对象是否结构化分为自然语言编码、数据库编码和自然语言与数据库之间的模式链接 3 类. 在自然语言的编码方面, 得益于自然语言处理领域的发展, 目前已有大量的研究工作, 如词嵌入和预训练模型等. 在数据库编码方面, 数据库内容可以细分为数据库模式和数据库记录. 另外目前也有研究关注单独提升模型对模式链接的捕获能力, 从而提升模型生成 SQL 的效果. 数据库编码涉及的相关技术主要包括循环神经网络、图神经网络和预训练模型. 解码模型可以根据生成 SQL 语句的不同方式分为基于模版插值的方法和基于生成的方法. 基于模版插值方法的特点是不要求生成完整的 SQL 语句, 只需要生成模版中需要插入的相关内容. 但基于模版的方法在面向复杂程度不同的 SQL 语句时具有较大区别, 所以后续将分别讨论. 基于生成的方法模型需要生成完整的 SQL 内容, 或者生成完整的等价表示形式. 在生成的方法中又可以根据生成 SQL 的形式不同分为基于序列生成的方法和基于语法生成的方法. 在基于序列生成的方法中, 模型将 SQL 语句看作是一种特殊的自然语言, 将 SQL 语句中的每一个元素看作是单词直接进行完整 SQL 序列的预测. 基于序列的方法通常也会搭配其他模块的优化来增强生成效果,

如强化学习和预训练模型等. 在基于语法生成的方法中, 模型直接预测 SQL 语句的抽象语法树, 相当于在生成过程中加入了语法的约束, 从而尽可能让模型学习到 SQL 语句的语法. 在基于语法生成的内部, 也包含了利用语法建模和针对语法优化两种思路, 第一种直接使用 SQL 语法约束解码过程, 第二种针对 SQL 的特点设计中间表示或使用其他语法作为中间表示来进一步优化解码过程. 编码器 – 解码器结构下的深度学习模型包含了编码模型和解码模型部分, 本文在分类时考虑文章中研究的侧重点进行展开, 例如仅在编码模型部分进行创新的文章, 将只会在编码模型部分总结, 不会在解码模型部分展开描述.

评估方法方面, 由于要求模型生成有约束的可执行 SQL 语句, 而不是无约束的自然语言, 所以对生成后的内容要求更高. 具体表现为首先要求生成的内容符合 SQL 语法, 而不是其他的形式, 另外还要求生成的 SQL 语句和标准答案的语义相同. 在 SQL 生成过程中, 常见的评估方法包括基于文本的组成成分匹配率和完全匹配率, 和基于执行的执行结果正确率. 实际中, 往往需要综合多个评价指标来评价生成 SQL 语句的质量. 另外, 也有研究不断探索如何更加客观地评价生成 SQL 的语义, 以及除了 SQL 语义以外其他评估指标.

2.5 相关综述

目前也有研究者从不同角度对 SQL 生成的工作进行整理和总结 [4, 5, 7]. Kalajdjieski 等 [4] 将现有关于 SQL 生成的研究划分为数据集、方法和评估指标 3 部分, 总结了目前常见的关于 SQL 生成的工作. 文献 [4] 首先在数据集方面, 整理和统计了常见的英文数据集. 然后, 在方法部分展示了多种基于循环神经网络 (recurrent neural networks, RNNs) 的深度学习模型. 最后介绍了相关的评价指标和各自的优缺点. 但文献 [4] 没有对现有 SQL 生成相关方法进行系统地分类, 只是罗列了 SQL 生成中常见的模型. 这导致对相关技术发展情况以及对主要创新点所属类别的描述不够清晰. 并且, 文献 [4] 主要总结了关于编码器 – 解码器 (encoder-decoder) 框架中涉及的循环神经网络和双向注意力机制相关技术, 缺少对编码过程中图神经网络 (graph neural network, GNN) 和预训练模型部分的相关文献总结, 但研究表明这些技术往往能够提升 SQL 生成的效果. 除此之外, 也有越来越多的研究关注中文相关 SQL 生成数据集的构建, 文献 [4] 也缺少对此的总结. Katsogiannis-Meimarakis 和 Koutrika [5] 发表了一篇基于深度学习的 SQL 生成方法教程. 首先介绍了关于 SQL 生成的两种数据集, 分别是 WikiSQL 和 Spider. 然后分为基于模版插值、序列到序列和基于语法的 3 种 SQL 生成方法进行介绍, 每种方法下主要涉及 2 种左右的具体文献介绍. 但文献 [5] 没有对评价指标和方法进行系统整理, 也没有对数据集相关研究进行分类和总结. 另外, 文献 [5] 主要基于教程的目的, 所以没有对与现有研究的相关技术进行深入探索, 也没有对目前在 WikiSQL 和 Spider 数据集上表现最好的一系列方法进行总结. 潘璇等 [7] 专门针对数据库自然语言接口进行讨论, 从解码类别的角度将现有研究分为 4 类, 分别是基于序列的生成方法、基于固定模板的生成方法、框架 – 细节的分阶段生成方法和基于语法的层级式生成方法. 针对这 4 类解码方法, 文献 [7] 也归纳了 7 种辅助类方法作为补充, 在补充的方法中既有涉及对编码阶段的处理, 也有对解码阶段的处理. 虽然文献 [7] 从编码器 – 解码器框架下进行分类, 但整体是从解码层面进行类别划分, 没有过多的关注 SQL 生成任务中编码器的特殊性. 文献 [7] 将编码阶段的创新工作归纳为辅助类方法, 没有将现有对编码阶段的研究工作划分在编码器 – 解码器框架下. 另外, 文献 [7] 的类型为系统性综述, 主要是对现有研究做了系统的整理, 没有具体分析当前研究的开展思路.

为了处理以上文献的局限性, 本综述首先建立了系统性的分类框架, 用于整理和分类现有研究工作. 分类框架根据 SQL 生成研究中创新点的侧重不同, 分别对编码器和解码器中的研究进行详细分

类. 然后在具体的数据和方法层面, 整理了目前 SQL 生成研究的具体工作和创新点. 具体而言, 全面整理了关于 SQL 生成研究的中文和英文数据集, 总结了涉及循环神经网络、图神经网络和预训练模型等一系列最新的编码模型研究, 以及基于模版、序列和语法的解码模型的研究思路. 最后本文总结了相关工作存在的挑战并展望了未来的研究方向.

3 SQL 生成场景

SQL 生成任务根据不同场景的需求具有不同的特点. 下面整理了典型应用场景下 SQL 生成任务中所包含的场景元素. 这些场景元素可以根据实际的需求进行组合, 从而构成各具特色的 SQL 生成研究任务.

根据数据的语言特点可以得到面向不同语种的场景. 典型的语种包括英文和中文, 现有 SQL 生成数据集大多面向英文查询问题和 SQL 语句, 但实际应用中, 对中文数据库的查询同样存在大量的需求. 另外, 对于中文数据库中的数据查询, 查询问题也可以有中文或英文两种选项. 在不同语种场景下, SQL 生成的效果会受到影响, 相关的研究需要针对不同语种场景进行特定的适配.

根据具体 SQL 语句的复杂程度可以将生成任务分为简单和复杂两种场景. 简单生成场景下 SQL 语句一般只包含 “SELECT”, “FROM” 和 “WHERE” 关键字, 而复杂生成场景下的 SQL 语句常常包含 “GROUP BY” 和 “ORDER BY” 等关键字, 以及嵌套的复杂查询条件. 相比较于只包含简单 SQL 语句的任务而言, 这些复杂的查询语句会给生成模型带来更大的挑战.

根据 SQL 语句所涉及数据表的复杂程度, 可以将生成任务分为简单和复杂数据表两种场景. 简单的数据表往往是单表的数据库, 并且数据表中的模式和数据记录接近自然语言的描述, 这样的数据表与自然语言的分布差异相对较小, 生成 SQL 的性能较高. 复杂的数据表通常涉及的数据表模式相对复杂, 数据表复杂性可以具体体现在两个方面, 首先是数据库模式之间的链接关系相对较多, 其次是数据模式的描述更加接近真实开发场景, 这就导致了数据表和自然语言的分布差异较大, 对模型的模式链接能力要求更高.

根据 SQL 生成过程是否存在上下文依赖, 可以将生成任务分为单轮 SQL 生成场景和多轮 SQL 生成场景. 常见的 SQL 生成任务属于单轮 SQL 生成场景下的应用, 其特点是根据一句自然语言描述生成一句对应的 SQL 语句. 多轮 SQL 生成场景可以允许进行多次的 SQL 生成, 并且当前 SQL 语句的生成可能依赖上次或者之前 SQL 语句的生成结果. 这种多轮的场景可以用于难以一次性描述清楚查询意图的情况, 以及当前查询需要参考之前查询结果的情况.

根据所涉及的 SQL 数据是否包含来自多个不同领域内的数据库内容, 可以针对数据库领域的场景进行 SQL 生成任务设计. 若一个数据集只包含一个领域内的知识, 那根据该数据集生成的 SQL 语句通常只能适配当前领域内的应用. 如果需要训练的深度学习模型能够将 SQL 生成泛化到未见过的数据库领域, 则需要收集多个领域内的数据集进行训练. 本文在对数据集综述方面首先将根据此类场景进行分类, 然后分别讨论每个类别下已有研究所涉及的其他场景.

在实际开发过程中, 常常会涉及除了 SQL 语句以外的数据库操作代码, 如数据库连接, SQL 查询语句预处理等使用 SQL 外部代码的场景. 这是一种 SQL 生成的拓展任务场景, 生成外部代码能够促进操作数据库相关工作的自动化, 并进一步提升实际开发场景中的 SQL 编写和使用效率.

现有的研究主要针对 SQL 语句复杂程度、数据库领域和多轮 SQL 生成场景进行了研究, 并提出了一系列的数据集和模型方法. 但面向不同语种和实际应用的 SQL 生成研究任务较少, 为了丰富 SQL 生成任务和应用场景, 针对不同场景的工作还有进一步研究的空间.

表 1 单领域数据集代表文献总结

Table 1 Summary of representative literature on sigle-domain datasets

Paper	Year	Dataset	#Num	Domain
Price [8], Iyer et al. [9]	1990	ATIS	5280	Air travel information
Zelle and Mooney [10], Popescu et al. [2]	1996	GeoQuery	877	Geographic information
Tang and Mooney [11], Popescu et al. [2]	2000	Restaurants	378	Restaurant, food and location information
Li et al. [12]	2014	Academic	196	Microsoft academic information
Yaghmazadeh et al. [3]	2017	Yelp	131	Yelp website
Yaghmazadeh et al. [3]	2017	IMDB	128	IMDB website
Iyer et al. [9]	2017	Scholar	817	Academic publication information
Finegan-Dollak et al. [13]	2018	Advising	4570	Student course information

4 SQL 生成数据集

构造大规模高质量数据集是基于深度学习的 SQL 生成研究的基础, 数据集的设计和构造也一直是 SQL 生成领域的一个关键研究问题. 早期关于数据集的研究主要针对特定领域的数据库收集查询问题和相应的 SQL 语句. 近年来, 更多的研究工作集中于跨领域 (cross-domain) 的 SQL 生成, 即针对多个数据库设置问题和相关的 SQL 语句, 这要求 SQL 生成模型在学习某些领域上的知识后能够泛化到不同领域的查询需求上. 本节将从数据集是否跨领域的角度对已有研究工作进行系统总结.

4.1 单领域数据集

在研究早期, 大部分 SQL 生成数据集只涉及了某个特定领域单个数据库. 这类数据集常常包含一个数据库中的多个数据表. 训练和测试过程所使用的数据均由该数据库产生. 表 1 [2,3,8~13] 展示了单领域数据集的代表文献和数据集基本统计数据. 可以发现单领域的数据集的构造时间较早, 相应的数据量也较少.

根据不同领域的具体场景和相应数据, 不同研究者对各种领域内的查询问题和 SQL 数据进行了单独的收集. Tang 和 Mooney [11] 以及 Popescu 等 [2] 先后整理得到了关于餐馆、食物以及相关位置信息的 Restaurants 数据集. Li 等 [12] 在微软学术搜索 (Microsoft Academic Search, MAS) 上收集了 Academic 数据集. 其中 SQL 语句通过枚举 MAS 网站的搜索页面表示的每个逻辑查询得出. Yaghmazadeh 等 [3] 收集了关于 Yelp 和 IMDB 网站上相关的查询数据, 构造了 Yelp 和 IMDB 数据集. 数据集中的问题均由数据标注者提出, 其中问题内容表达了标注者希望在 Yelp 和 IMDB 数据库进行查询的意图. Iyer 等 [9] 基于原始关于航班预订的 ATIS 数据集 [8,14], 将原来低效率的 SQL 查询进行了修改, 构造了新版本的 SQL 生成数据集. 在此基础上, 进一步验证了修改后的查询结果没有改变, 保证数据集功能的一致性. 同时在 Iyer 等的这篇工作中, 他们也发布了自己的数据集 Scholar, 其中收集的数据为用户关于学术发表信息的查询问题. Scholar 与 Academic 数据集类似, 但使用的数据库存在差异. Popescu 等 [2] 和 Iyer 等 [9] 先后标注和完善 GeoQuery 数据集. GeoQuery 是美国地理相关问题的查询数据, 原始版本为查询问题和逻辑编程语言组成的数据集, 主要用于语义解析任务. 后来修改成为查询问题和 SQL 语句的数据, 可以用于 SQL 生成任务. Finegan-Dollak 等 [13] 提出 Advising 数据集, 这是关于 Michigan 大学课程信息的数据库查询数据集. 其中一些查询问题是从相关部门的 Facebook 页面收集的, 其他查询问题是由了解数据库的计算机专业学生编写的. 这篇工作中, Finegan-Dollak 等还对之前的 SQL 生成相关数据集进行了分析和处理, 主要包括对数据的去重和修正处理. 最后发布

表 2 跨领域数据集代表文献总结

Table 2 Summary of representative literature on cross-domain datasets

Paper	Year	Dataset	#Num	#DB.Num
Zhong et al. [15]	2017	WikiSQL	80654	24241
Shi et al. [16]	2020	SQUALL	11276	2108
Yu et al. [17]	2018	Spider	9693	200
Lee et al. [18]	2021	KaggleDBQA	272	8
Min et al. [19]	2019	CSpider	9691	166
Sun et al. [20]	2020	TableQA	49974	5291
Wang et al. [21]	2020	DuSQL	23797	200
Yu et al. [22]	2019	SParC	12726	200
Yu et al. [23]	2019	CoSQL	15598	200
Guo et al. [24]	2021	CHASE	17940	280

了关于之前 SQL 生成研究工作数据集的处理后版本, 为后续的 SQL 生成任务提供了便利.

目前单领域的数据集主要使用英语描述和英文数据库, 并且采用单轮的方式生成简单的 SQL 语句. 单领域的 SQL 生成数据集只要求 SQL 生成系统学习某个特定领域的知识, 并在该领域的类似问题上进行应用. 这类数据集适合训练任务目标单一的 SQL 生成模型, 这类模型通常应用范围有限且无法泛化到其他领域的 SQL 生成任务.

4.2 跨领域数据集

在真实应用场景中, 数据库查询任务常常涉及多个数据库, 因此要求 SQL 生成模型能够泛化到训练过程中没有出现过的数据库上. 表 2 [15~24] 总结了跨领域的数据集的代表文献和统计数据. 跨领域数据集将多个领域的数据库进行了汇总, 要求模型在其中一些数据库领域上训练, 并泛化到不同领域的数据库上. 与单领域的数据集相比, 跨领域的数据集涉及的数据库和数据数量也更多.

跨领域数据集往往相较于单领域数据集有更大的规模, 但跨领域数据集内部的 SQL 语句复杂度也是区分跨领域数据集的特征之一. Zhong 等 [15] 从 Wikipedia 中抽取了 24241 个 HTML 数据表, 使用亚马逊众包平台进行人工标注, 构造了大型的 WikiSQL 数据集. 标注过程中首先使用模版的方式生成粗粒度的查询问题, 然后让标注者根据问题和数据表写出相应的 SQL 语句. 后续也对问题描述和 SQL 语句的一致性进行了人工检验. WikiSQL 包含了多个领域内的单表数据库, 但其中的 SQL 语句相对简单, 没有涉及复杂的 SQL 操作. Shi 等 [16] 基于 WikiTableQuestions [25] 中相关查询构造了 SQUALL 数据集, 其中查询问题不仅包含简单的表格查找, 还具有高度的组合性. 在 SQUALL 数据集中, 首先根据查询问题构造了目标代码的 SQL 表示形式, 然后标注了查询问题和目标 SQL 之间的词汇映射关系. 这种细粒度的关联信息减少了模型学习输入输出之间对齐信息的难度, 实验结果也证明了这种标注信息的有效性. Yu 等 [17] 构造了一个大规模、复杂的跨领域 SQL 生成数据集 —— Spider. 该数据集包含了 200 个数据库, 183 个不同的领域, 以及非常复杂的 SQL 查询语句. Spider 数据集中 SQL 复杂的原因主要包含两方面: (1) 查询当前表的条件里可能需要从其他数据表获取信息; (2) SQL 语句中包含了复杂的查询条件, 如 “GROUP BY” 和 “ORDER BY”. 相对于 WikiSQL, Spider 数据集明显提升了对模型生成 SQL 能力的要求. Lee 等 [18] 基于 Kaggle 竞赛平台上真实的数据库收集了跨域评估数据集 KaggleDBQA. 为了符合工业界真实的场景, KaggleDBQA 中的数据库没有对数据库模

式进行额外的预处理, 查询问题中也尽量避免了提到数据库中的列名称. KaggleDBQA 同时也使用了数据库文档和相关描述来增强模型对领域内信息的学习, 以此增加在领域外预测的正确率. 与 Spider 相比较, KaggleDBQA 在 SQL 语句上具有更多比例的高难度查询, 并且更加符合工业界生成 SQL 的场景.

区分跨领域 SQL 生成数据集的另一个特征是语言类型, 之前的大多数据集都使用英语作为默认语言, 但为了适配使用中文的应用场景, 也有一些研究关注中文 SQL 数据集. Min 等 [19] 根据 Spider 数据集, 将英文描述通过人工翻译成中文描述, 从而构建了 CSpider. 这种类型的数据集给 SQL 生成带来了如下两点新的挑战, 首先是建立中文查询问题和英文的数据表的映射存在困难, 其次是以单词为单位对数据表进行分词可能出错. 在此基础上, Min 等也通过 CSpider 验证了跨语言 (cross-lingual) 的词嵌入方法的有效性. Sun 等 [20] 从不同领域收集了超过 6000 个数据表进行标注和检验, 构造了中文数据集 TableQA. 与之前的工作相比较, TableQA 考虑到了真实场景中同一个事物可能会有不同的表达, 以及查询的问题是否可以回答两个常见的问题. TableQA 数据集中的 SQL 复杂程度与 WikiSQL 类似, 但使用的数据库是中文数据库. Wang 等 [21] 构造了一个大型的跨领域 SQL 生成的数据集 DuSQL, 其中使用中文描述和中文数据库. 该文首先分析了用户在真实场景中使用 SQL 查询语句的分布, 并考虑了其中大量的数据表中行/列运算的情况. 接着在数据构造过程中, DuSQL 通过定义的 SQL 语法自动生成 SQL 查询语句和对应的伪语言问题描述, 并通过众包方式将伪语言问题描述改写为自然语言问题.

近年来, 在跨领域数据集的基础上, 上下文依赖的多轮 SQL 生成任务也引起了越来越多研究者的关注. 这类 SQL 生成任务适用于需要向 SQL 生成系统询问一系列问题来进行复杂查询的场景, 要求 SQL 生成模型能够捕捉前文查询的信息并用于后续的查询过程. Yu 等 [22] 构造了一个包含 4298 轮对话的 SQL 生成数据集 SParC, 该数据集不仅要求模型能够泛化到新的领域, 并且要求从丰富的上下文交互中学习其中的依赖关系. SParC 中使用原始 Spider 数据集中的查询问题作为构建交互查询的指导信息, 数据标注者根据 Spider 中的查询问题提出相互关联的问题并作为交互的目标. 数据集中的交互问题根据复杂的 SQL 查询改编, 并由标注者进行一定程度的交互探索, 最后标注者将每一个交互中的问题转化为 SQL 查询语句, 从而构建此类数据集的数据单元. 2019 年, Yu 等 [23] 继续提出一个包含 3007 轮对话的 SQL 生成数据集 CoSQL. 相比较于之前的 SParC, CoSQL 的数据更加贴近真实场景中向 SQL 生成系统询问的情况. 当向 SQL 生成系统查询时, CoSQL 数据集中考虑了如果无法用 SQL 语句进行回答时, 应当以一定的方式向提问者澄清或提示该问题无法回答, 如果能够回答则返回相应的 SQL 查询语句. Guo 等 [24] 收集了一个包含 5459 轮对话的复杂中文数据集 CHASE. 与之前的跨领域上下文依赖数据集相比较, CHASE 增强了上下文依赖的特点以及增加了对话过程中 SQL 的复杂程度. CHASE 可以分为两部分, CHASE-C 和 CHASE-T. 在 CHASE-C 中, 12 名学生作为标注者进行问题序列的建立, 以及相应 SQL 语句的标注. 在此过程中还提供了查询意图推荐方法来保证多样性. 在 CHASE-T 中, 类似于 CSpider, 直接将 SParC 中的交互查询数据集翻译为中文数据集并缓解中英文之间的偏差. 从而将两部分结合构建了第一个大型中文跨领域上下文依赖的 SQL 生成数据集.

4.3 小结

从单领域的数据集到跨领域数据集, 再到跨领域上下文依赖的数据集, 随着 SQL 生成技术的发展, SQL 生成相关数据集也变得越来越复杂, 越来越贴合实际的应用场景. 与此同时, 也有越来越多的研究者关注大规模中文 SQL 生成数据集的研究, 这有利于吸引研究者开展中文描述和数据库相关编码的研究. 数据集是基于深度学习的方法中非常重要的一部分, 数据难度的增加给 SQL 生成领域不断

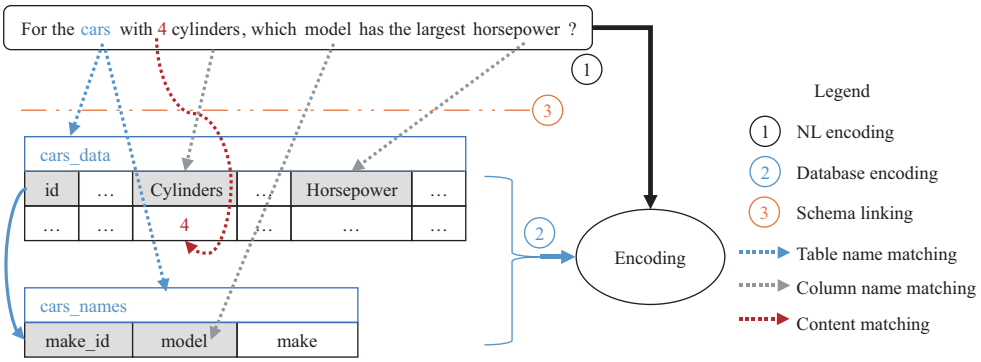

图 3 编码过程示例

Figure 3 An example of encoding process

提出新的挑战, 推动了 SQL 生成相关方法的研究, 使得基于深度学习的 SQL 生成模型一步步向实际应用迈进. 目前在现有某些数据集上, 最先进的方法达到非常高的准确率, 而剩下的情况可能由标准的 SQL 语句标注错误导致. 因此我们应注意到数据集中标注偏差带来的误报率, 检测数据集中查询语义与 SQL 语句不对应的情况, 提升数据质量. 另外, 现有数据集通常要求人工参与标注和检查, 因此大规模数据集往往意味着大量的标注代价. 后续研究应继续探索面向实际应用场景的高质量数据集, 以及如何降低对大规模数据集的标注开销.

5 模型结构

5.1 编码模型

基于深度学习的方法首先要求模型的输入为数字类型的向量, 并要求编码后能够尽可能地保留原始输入的所有语义. 将输入的信息编码为合适的向量形式是训练深度学习模型中非常重要的环节, 它将直接决定后续语义解析的效果. 由于 SQL 生成的特殊性, 在编码阶段需要同时考虑自然语言形式的查询问题、结构化的数据库信息以及自然语言与数据库之间的模式链接信息.

图 3 展示了编码过程中需要考虑的内容. 其中编号 1 为自然语言编码部分, 编号 2 为数据库编码部分, 编号 3 为模式链接部分. 如图 3 所示, 数据库相关的编码不仅需要考虑数据库模式的编码, 即针对 cars_data 和 cars_names 两张数据表的编码, 还需要考虑模式链接的编码, 例如将自然语言中的 cars 对应到图中的两张表, 将 cylinders, model 和 horsepower 对应到表中的列名称. 现有研究工作包括了在数据库编码部分考虑模式链接的做法, 以及单独考虑提升模型对模式链接理解能力的做法, 下面将对近年来专注于提升模型模式链接能力的研究工作分开讨论. 除此之外, 数据库内容可以分为数据库模式和数据库记录, 一些研究工作也对数据库中的记录进行编码, 例如将自然语言中的数字 4 对应到数据表中的记录.

5.1.1 自然语言编码

SQL 的查询问题通常以自然语言的形式描述, 如何在模型中表示此类自然语言的描述也是自然语言处理 (natural language processing, NLP) 领域的重要研究方向. 随着深度学习的发展, 目前基于神经网络的词分布表示, 即词嵌入 (word embedding), 是生成自然语言表示的最有效方法. 词嵌入模型往往通过训练语言模型 (language model) [26] 或者使用特定的神经网络 [27] 来生成单词和其上下文的

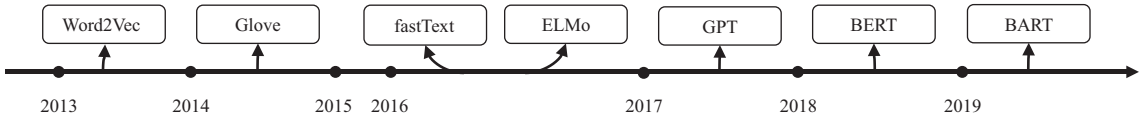

图 4　经典词嵌入模型的时间线

Figure 4　A timeline of classical word embedding models

语义表示. 图 4 展示了在词嵌入领域具有重要影响的经典模型.

Word2Vec [28, 29] 和 Glove [30] 是词嵌入较早时期的经典模型. Word2Vec 使用连续词袋模型 (continuous bag-of-words, CBOW) 和跳字模型 (skip-gram) 来捕捉单词的语义信息并将它们转化到下游任务使用. 其中连续词袋模型: 输入已知上下文, 输出对下个单词的预测. 与此相反的是跳字模型: 输入某个单词, 输出对它上下文词向量的预测. 这两种方法从已有语料中训练后可以得到每个词的表示信息. Glove 在 Word2Vec 的基础上又利用共现矩阵考虑了全局和局部的信息, 以此来弥补 Word2Vec 只考虑局部窗口所带来的不足. fastText [31] 是一个计算词向量和文本分类的工具, 与之前的工作相比较, 它并没有将单词作为不可分割的单位, 而是考虑了字符级别的 n-gram 的信息. 这样带来的好处是能够对词表之外的单词计算词向量, 因为它们的 n-gram 可能与词表内的单词共享.

然而, 以上词嵌入方法得到单词向量后往往需要经过特定的神经网络才能得到整体的自然语言编码表示, 这类特定的神经网络包括了 RNN [27] 系列网络和 Transformer [32] 系列网络. 传统神经网络计算隐藏层输出如式 (1) 所示:

$$\boldsymbol{H} = \phi(\boldsymbol{X}\boldsymbol{W}_{xh} + \boldsymbol{b}_h), \tag{1}$$

其中 $\boldsymbol{X}$ 为输入的批量数据, ϕ 为激活函数, $\boldsymbol{W}_{xh}$ 和 $\boldsymbol{b}_h$ 为可学习参数, $\boldsymbol{H}$ 为隐藏层输出. 如果将上述隐藏层用于输出, 则只需增加式 (2):

$$\boldsymbol{O} = \boldsymbol{H}\boldsymbol{W}_{hq} + \boldsymbol{b}_q, \tag{2}$$

其中 $\boldsymbol{O}$ 为输出结果, $\boldsymbol{W}_{hq}$ 和 $\boldsymbol{b}_q$ 为可学习参数. 与传统神经网络相比, RNN 增加了隐藏状态的特性, 如式 (3) 中的 $\boldsymbol{H}_{t-1}$ 所示:

$$\boldsymbol{H}_t = \phi(\boldsymbol{X}_t\boldsymbol{W}_{xh} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hh} + \boldsymbol{b}_h). \tag{3}$$

计算当前时间步骤 t 的隐藏层状态 $\boldsymbol{H}_t$ 需要利用 $t-1$ 步的隐藏层状态 $\boldsymbol{H}_{t-1}$, 其中新增的 $\boldsymbol{W}_{hh}$ 为可学习参数, 用于描述如何使用上一个隐藏层状态. 同样地, 时间步骤 t 的隐藏层状态会被用于下一时间步骤隐藏层状态的计算, 所以是具有循环特点的神经网络. 具有这种特点的神经网络可以捕捉序列前后的联系, 更有利于序列信息的表示, 目前也有一系列研究基于 RNN 结构进行改进和应用 [33~37]. Tranformer 是区别于 RNN 系列模型的另一种网络结构, 其利用自注意力机制可以缓解 RNN 系列模型捕获文本长期依赖能力不足的问题. 与 RNN 逐个处理单词不同, 自注意力机制中每个单词都通过自注意力连接到其他单词, 具有并行的特点. 自注意力机制因为并行计算而放弃了顺序操作, 并为输入添加位置编码来表示绝对和相对的位置信息. Transformer 模型还希望能够通过同一种注意力机制学习到不同行为, 所以采用多头注意力机制组合使用不同的查询、键和值的不同表示. 多头注意力机制中具体的第 s 个头注意力计算过程如式 (4) 所示:

$$\text{head}_s = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{4}$$

其中 d_k 表示一个头注意力中的模型维度, $\boldsymbol{Q}$, $\boldsymbol{K}$ 和 $\boldsymbol{V}$ 表示由输入计算而来的查询、键和值. 在自然语言处理的多项任务中, Transformer 相比 RNN 表现出了更强的能力, 并且后续优秀的词嵌入模型往往

基于 Transformer 中的编码模型改造而来. 值得注意的是这类特定的网络结构通常也会被用于处理多轮对话的编码, 如 Sordoni 等 [38] 提出的层次循环编码 – 解码器 (hierarchical recurrent encoder-decoder, HRED) 模型, 采用 RNN 模型分别对当前会话和之前上下文进行编码, 并且将两者结合后进行解码. Zhang 等 [39] 提出 Recosa 模型, 同时将 RNN 系列模型的编码和 Transformer 中的自注意力机制引入到多轮对话的编码.

为了更好地利用大规模数据和特定神经网络结构的优势, 使得编码模型能够识别相同单词在不同语境下的具体含义, 一系列基于 RNN 和 Transformer 的预训练自然语言编码研究也随之提出. 其中 ELMo 模型 [40] 通过双向 LSTM 学习单词的表示并在大量语料上预训练相应的语言模型, 期望模型能够学习到单词语法和语义的复杂用法以及这些复杂用法在不同上下文的变化. 实验证明 ELMo 在下游任务微调能够在多种语义的表示上取得较好效果. 不同于 ELMo, GPT [41] 使用 Transformer 的解码器为主要结构来进行单词表示的学习, 训练过程中采用语言模型的经典思路, 即通过前几个词来预测后一个词. BERT [42] 与 ELMo 的工作方式类似, 但选择使用双向的 Transformer 编码器为基础结构来捕捉深层语义信息. 不同于之前使用从左往右进行单词预测的方式, BERT 使用了掩码语言模型 (masked language model, MLM) 和下一句话预测 (next sentence prediction, NSP) 两个子任务来进行预训练. MLM 任务用于预测句子中被遮住的 15% 的单词, NSP 任务用于判断两个句子是否连接在一起, 两者结合可以使得 BERT 捕捉句子内部和句子之间复杂的语义信息. BART [43] 采用了 Transformer 的整体结构为基础框架, 包含了双向的编码器和从左到右的自回归解码器, 具有 BERT 和 GPT 模型的特点. 预训练包含两个部分, 首先通过噪音函数破坏输入的文本, 然后通过学习让模型重新恢复和构造原始输入. BART 通过恢复被打乱的文本能够让模型捕获更多文本内部联系, 从而在文本生成类任务中获得更好的表现. 同样地, 预训练模型, 如 BERT, 也可以用于多轮对话的生成 [44].

后续仍然有大量的工作基于这些经典模型进行改进, 如 GPT2 [45], GPT3 [46], RoBERTa [47], CodeBERT [48] 等. 这些研究工作极大地促进了 NLP 领域相关任务的研究, 也对于 SQL 生成任务的自然语言编码部分提供了大量的思路和工具.

5.1.2 数据库编码

数据库的编码是 SQL 生成研究任务的特色之一, 在早期单领域的数据集中由于描述较为详细且 SQL 语句相对简单, 不进行数据库编码处理也能得到较好的效果, 但随着数据集的跨领域以及 SQL 语句复杂性的提升, SQL 生成越来越要求模型编码过程中能够更好地处理结构化的数据表信息. SQL 生成任务中所涉及的数据库为关系型数据库, 以行和列的形式组织存储数据. 数据库的信息可以细分为数据库模式 (database schema) 和数据库记录, 数据库模式指列名称、类型和外键等数据表逻辑结构, 数据库记录指数据表中的数据记录. 一般来说, 生成 SQL 查询语句过程中往往需要使用数据表的逻辑结构, 但也有研究表明使用数据库中的数据记录可以进一步增强模型的生成效果. 本小节将总结和讨论对数据库模式和内容编码的相关研究.

数据库编码可分为数据库单独编码和自然语言数据库联合编码两种方式. 图 5 展示了自然语言和数据库经典的编码思路, 其中下方两种思路为数据库单独编码方式, 上方为自然语言和数据库联合编码的方式. 第一种思路是将数据表和查询问题转化为序列后使用可编码序列的神经网络模型进行编码, 如使用循环神经网络系列模型. 第二种思路是使用图神经网络对数据库的内部结构进行编码, 然后与自然语言的编码结合后得到最终的编码. 第三种思路是使用预训练模型对自然语言和数据库信息编码. 图中预训练模型以 BERT 为例, 编码考虑了输入的单词、自然语言与数据库信息的分隔, 以及位置信息. 编码过程使用特殊字符将多个数据表、列名称以及值进行区分, 以此尽可能地捕捉数据库

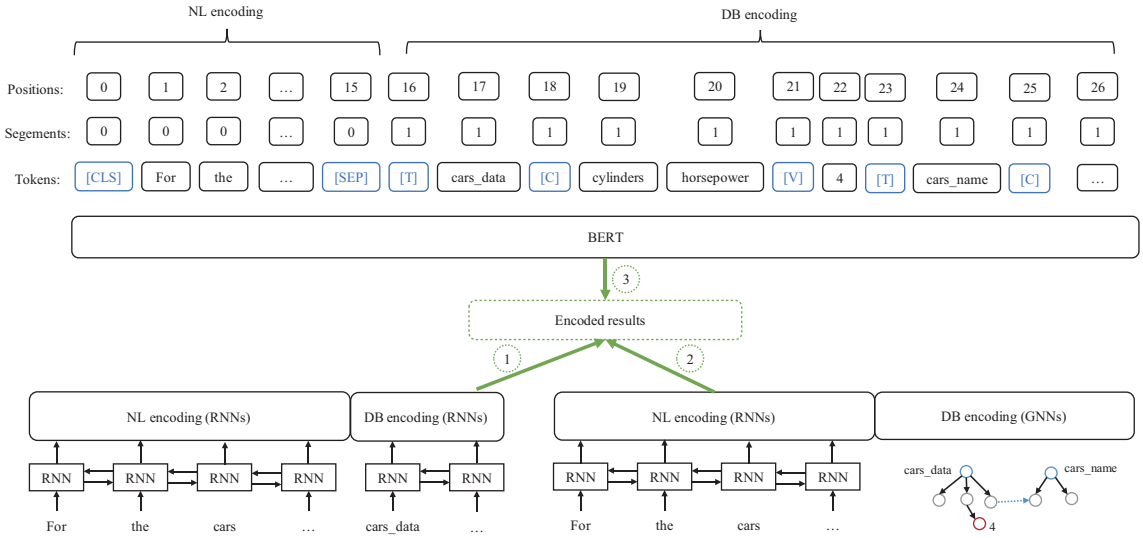

图 5　(网络版彩图) SQL 生成中经典的数据库编码方法

Figure 5　(Color online) Classic encoding methods in SQL generation

内部的联系.

数据库单独编码方式下, 模型往往需要单独对自然语言和数据库进行编码, 并使用注意力机制将两者的编码进行联合. 在数据库编码的早期研究中, Xu 等 [49] 在 SQLNet 的研究工作中使用了列注意力 (column attention) 机制来增强模型捕捉查询问题和列名称之间的联系, 其特点是增加注意力机制来增强模型捕捉查询问题和列名称之间联系的能力. Yu 等 [50] 提出了 TypeSQL, 利用列类型和模式链接等先验知识辅助编码过程. 在具体类型的计算过程中, 首先将查询问题分词后与数据库模式中的单词进行匹配, 并将能够匹配列名称的单词作为列类型. 接着将时间、数字和命名实体根据先验知识进行识别和标记. 由于表的内容是可以获取的, 所以也将查询问题和表内容进行匹配以此来标记查询问题中的列类型. 实验结果证明使用类型信息可以更好地处理查询问题中罕见的实体和数字内容.

随着图神经网络的发展, 结构化的数据库信息也逐渐使用图神经网络进行编码. Bogin 等 [51] 使用图神经网络 GNN-SQL 来编码数据库模式的结构, 这种 GNN 的编码形式也同时在编码和解码阶段使用, 以此让模型在未见过的数据库模式上具有更好的泛化性能. 使用 GNN 编码数据库模式的过程中, 将数据库中的表、相关的列以及表之间的关系表示为一张图的形式, 并使用与查询问题相关的子图来丰富查询问题中每个单词的表示. 2019 年, Bogin 等 [52] 继续提出 Global-GNN 网络来解决对输出查询的结构进行全局推理的问题. 该模型使用全局门控 (global gating) 机制, 与之前使用局部信息的方法相比较, 能够进一步提升 SQL 生成的性能. Wang 等 [53] 注意到之前的工作在数据库模式编码和模式链接中以单独的特征化技术来增强单词向量的表示, 而不是编码单词和列之间的关系. 所以提出了 RAT-SQL 框架, 提供了一种统一的方法来编码输入之间的任意关系信息. 其特点是利用一个完整的关系图注意神经网络来处理各种预先定义的关系, 其中关系感 (relation aware) 机制允许编码查询问题单词和数据库模式元素之间的任意关系, 并且这些表示是使用自注意机制在所有输入上联合计算的. Scholak 等 [54] 在 RAT-SQL 的基础上提出了 DuoRAT, 与构建越来越复杂的编码结构不同, DuoRAT 模型主要关注 RAT-SQL 模型有哪些部分可以简化. 该研究证明基于启发式的模式链接带来的好处优先, 而基于内容的模式链接和关系模式的链接相对更加重要. Chen 等 [55] 提出 ShadowGNN, 首先使用图映射网络去掉自然语言和数据库模式的语义信息, 得到抽象的结构, 接着使用 RAT 框架来获得整体

的编码, 以此来缓解特定领域信息的影响, 让模型具有更好的跨领域能力. Cao 等 [56] 提出了 LGESQL 来考虑边的拓扑结构, 提出了一种增强线图 (line graph), 从原始以节点为中心的图中构造了以边为中心的图, 并利用这两种图分别捕捉节点和边的拓扑结构. 借助于线图, 消息不仅通过节点之间的连接, 而且通过有向边的拓扑结构进行更有效的传播, 进而提高了编码能力.

自然语言和数据库联合编码往往需要使用训练模型来捕获两者之间的上下文关联信息. Hwang 等 [57] 在 SQLova 模型、He 等 [58] 在 X-SQL 模型、Lyu 等 [59] 在 HydraNet 以及 Lei 等 [60] 在 SLSQL 的研究工作中均使用 BERT 预训练模型编码查询问题和涉及的数据库模式. Zhang 等 [61] 在 EditSQL 工作中提出了查询问题和数据表两者之间的 co-attention 机制并使用 BERT 作为编码工具. 在查询问题编码中, 引入数据表中的列名称进行注意力机制计算, 以此获取最相关的列. 在数据表的编码中, 首先使用自注意力机制捕捉内部结构, 然后使用注意力机制捕捉查询问题和数据表模式的关系. 两种编码因 co-attention 机制而相互依赖, 因此随着查询问题的变化, 列名称的编码也会变化. Lin 等 [62] 提出了 BRIDGE 模型, 在使用 BERT 预训练模型对数据库模式编码的基础上, 又使用模糊匹配的方式从相关数据库中选取相应列中的记录, 并利用数据库中的记录来进一步增强列的表示. Ma 等 [63] 提出 IE-SQL 模型, 利用序列标注的方法抽取需要插值处的意图和关系, 并使用文本匹配的方法进行模式链接, 进一步提升了在 WikiSQL 数据集上的 SQL 生成性能. Zhong 等 [64] 提出了 GAZP 框架, 利用合成新环境下的数据来增强模型编码的学习过程. 不同于常见的数据增强方法, GAZP 使用前向的语义解析器和反向的查询问题生成器生成数据并进行一致性验证. GAZP 在 Spider, Sparc 和 CoSQL 数据集上进行实验并提升了性能, 证明了提升的性能与数据合成数量成比例, 并且数据的循环一致性是关键.

也有研究针对 SQL 生成任务的特点, 重新预训练特定于 SQL 生成任务的模型. 表 3 展示了 SQL 生成任务中具有代表性的预训练模型, 其中前 3 种模型针对 NLP 领域任务, 使用纯自然语言数据进行预训练, 后 5 种模型针对 SQL 生成任务的特点进行了适配和优化. 表中列出了这些经典预训练模型的名称、预训练使用的数据集、预训练目标以及所参考的基础模型, 其中前 3 种自然语言处理领域的预训练模型参考 Transformer 的网络结构, 后 5 种预训练模型参考自然语言处理领域已有的预训练研究. Yu 等 [65] 提出 GraPPa 预训练方法, 以 RoBERTa 为基础并进一步在生成数据上进行预训练. 其中大量的生成数据通过同步的上下文无关语法构造. 预训练目标为掩码语言模型和 SQL 语义预测 (SQL semantic prediction, SSP). 其中 MLM 用于预测被遮掩的单词, SSP 用于预测某列是否出现在 SQL 语句中以及使用的聚合操作. 实验证明了 GraPPa 在多个数据集上超过了 RoBERTa 模型. Yin 等 [66] 基于英文维基百科和 WDC 网页中的 2600 万个数据表 [67], 以及数据表中的上下文训练了 TaBERT 模型. 在自然语言表示的学习中使用 MLM 模型, 在列表示学习中使用列遮掩和单元值恢复作为预训练目标. 实验结果表明, TaBERT 可以在下游任务上的 WikiTableQuestions 和 Spier 数据集上表现出了较好的性能. Shi 等 [68] 提出了一种生成增强的预训练框架, 用于解决现有通用预训练框架面临的 3 个问题: 无法识别查询问题中提及的列, 无法从单元格值推断提及的列, 以及无法组合复杂的 SQL 查询. GAP 使用 4 个预训练目标, 除了MLM 以外, 还有列预测、列恢复, 以及 SQL 生成. 其中列预测学习的目标是通过预测一个列是否被用于话语中, 列恢复学习的目标是通过基于采样的单元格值恢复列名, 增强了模型发现单元格值和列名之间的链接的能力, 而 SQL 生成学习的目标是直接与下游任务相关. GAP 所使用的数据来源于 Github 以及由 Spider 上数据表生成. 最终模型在 Spider 数据集上进一步提升 SQL 生成性能. Xuan 等 [69] 提出了一种通过数据库模式感知降噪的方法, SeaD, 来预训练基于 Transformer 的模型. 模型主要特点是使用预训练模型来训练 Erosion 和 Shuffle 两个降噪目标. Erosion 针对表格内的成分进行变换, 包括了对列的重新排序, 以特定概率删除某列或从其他数据表中选则列增加到当前表中. 此时相应的 SQL 也应当改变, 以此来让模型学习到自然语言的查询问

表 3　SQL 生成任务中具有代表性的预训练模型

Table 3　Summary of representative pre-trained models for text-to-SQL task

Model	Pre-training dataset	Pre-training objectives	Basic model
BERT	BooksCorpus, English Wikipedia	(1) Masked language model (2) Next sentence prediction	Transformer-encoder
RoBERTa	BooksCorpus, English Wikipedia, CC-NEWS, OPENWEBTEXT, STORIES	Masked Language Model	Transformer-encoder
BART	BooksCorpus, English Wikipedia, CC-NEWS, OPENWEBTEXT, STORIES	(1) Bidirectional encoder (2) Autoregressive decoder	Transformer
Grappa	Synthetic Examples (WIKITABLES, Spider and WikiSQL), Natural Language Utterances (TabFact, LogicNLG, LogicNLG, HybridQA WikiSQL, WikiTableQuestions, ToTTo, Spider)	(1) Masked language model (2) SQL semantic prediction	RoBERTa
TaBERT	English Wikipedia, WDC WebTable Corpus	(1) Masked Language Model (2) Masked column prediction (3) Cell value recovery	BERT
GAP	English Wikipedia, SQL from GitHub	(1) Masked language model (2) Column prediction (3) Column recovery (4) SQL generation	BART
STRUG	ToTTo	(1) Column grounding (2) Value grounding (3) Column-value mapping	BERT
SeaD	WikiSQL	Schema-aware denoising (erosion and shuffle)	BART

题如何与数据库相连接, 以及在查询列不存在的情况下抛出异常. Shuffle 是将输入信息的内部实体打乱, 要求模型重新构造出一个正确顺序的输入. Shuffle 训练的目的是让模型能够捕获不同查询实体之间的内部联系. SeaD 通过预训练两个降噪目标来对结构数据更好地建模, 从而有效地提升了 SQL 生成效果.

5.1.3　模式链接

模式链接指将自然语言的单词映射到具体的数据库模式, 要求模型能够理解自然语言语义并与数据库模式进行匹配. 已有研究证明在 SQL 生成任务中模式链接是影响最终生成效果的关键因素 [60]. 目前也有一系列的研究工作关注模式链接, 包括提供更细粒度标注的数据集和专注模式链接能力的方法. 这些研究专注于单独增强模型捕获模式链接的能力, 并将这种能力用于下游 SQL 生成任务, 进而提升 SQL 生成的效果.

为了让模型具有更强的模式链接能力, 一些研究工作标注了自然语言和数据库模式之间的链接信息, 以此使用监督学习的方式使得模型具有更强的处理模式链接的能力. Lei 等 [60] 在 Spider 数据集的基础上, 标注了训练集和验证集上模型链接的详细信息, 构造了 Spider-L 数据集. Shi 等 [16] 在构造 SQUALL 数据集过程中同样提供了细粒度的模式链接信息, 方便模型能够更容易地学习到自然语

言中语义所对应的数据库模式. 细粒度的模式链接标注信息能够简化模型学习链接信息的困难, 也可以作为基准数据集测试不同方法表示模式链接的能力.

然而, 对模式链接信息的标注往往需要付出较大的代价, 通常这种粒度的链接信息也难以获得. 在没有对模式链接进行细粒度标注的情况下, 则需要相应的方法识别模式链接信息. 模式链接的方法可分为两种, 即基于字符串匹配的方法和基于语义的方法. 基于字符串匹配的方法根据自然语言和数据库模式之间的字符串匹配关系来判断自然语言中所指的数据库, 其中包括了完全匹配和使用 n-gram 模糊匹配两种具体方法. 但基于字符串匹配的方法缺少灵活性, 难以发现近义词之间的对应关系. 基于语义的方法根据神经网络计算自然语言和数据库模式之间语义差异来判断模式链接的关系. 简单的做法可以根据自然语言编码模型计算自然语言和数据库模式之间的语义信息, 然后再计算两种语义表示之间的相似度来判断链接关系. 也有一系列的研究工作单独学习模式链接信息用于强化 SQL 生成模型的效果. Lei 等 [60] 在标注了 Spider 上模式链接信息之后扩展了编码模型进行学习, 主要增加模式链接信息的监督式学习和感知模式的表示, 实验结果证明了模式链接学习能够在 Spider 数据集上提升效果. Liu 等 [70] 提出了 ETA (erasing-then-awakening) 方法, 首先训练一个概念预测模型计算数据库中模式是否被查询问题提及的可信度, 然后在预训练模型中通过遮掩查询问题的单词来获得可信度的变化, 并以变化量作为对应单词的重要程度, 最后使用预训练模型结合预测模块生成相应的模式链接关系. 实验结果证明 ETA 模型生成的模式链接关系能够被人类专家理解, 并且在 SQL 生成任务上能够达到较好的效果. Deng 等 [71] 提出 STRUG (structure-grounded) 预训练模型, 包括了列匹配 (column grounding)、值匹配 (value grounding) 和列 – 值匹配 3 种预训练目标. 前两个目标分别要求模型能够学习到如何将自然语言描述中的单词对应到数据表模式和数据记录, 而第 3 个训练目标用于处理列和值之间的对应关系. STRUG 可以有效捕捉模式链接信息, 并以此提升下游 SQL 生成任务的效果, 最终在 WikiSQL 和 Spider 数据集上表现出了较好的效果.

5.2 解码模型

深度学习模型在对输入数据进行深层次编码处理与计算之后, 需要将模型的内部表示形式解码为目标形式, 在 SQL 生成任务中的目标形式就是 SQL 语句及其等价的表示形式. 按照 SQL 生成方式的不同可以将解码模型分为基于模版插值的方法和基于生成的方法, 在基于生成的方法中又可以根据解码过程中是否有语法约束分为基于序列的 SQL 生成方法和基于语法的 SQL 生成方法.

图 6 展示了 SQL 语句生成方式. 第一种是基于模版插值的方法, 黑色框内的是需要插入的内容, 如果 AGG 表示聚合操作符号, COL 表示查询的列, TBL+ 表示查询的表, 可以由多个表连接而来, VAL 表示具体的数值类型. 第二种方式基于生成的方法, 可以具体分为是基于序列生成的方法和基于语法生成的方法. 基于序列生成的方法将 SQL 语句看作是由普通单词构成的序列, 并根据已有的序列预测下一个 SQL 中的单词, 挑选前 K 个概率最大的单词作为候选的下一个单词. 基于语法生成的方法是在 SQL 语句生成过程中加入 SQL 抽象语法树结构的约束, 一般以从上至下的方式生成, 即从树结构中的根节点往叶子结点 (终结符) 的方向生成.

表 4 [15, 50, 53, 56, 60, 69, 72~74] 展示了在 SQL 生成任务中的经典模型以及目前最新模型的技术特点. 其中 WikiSQL 和 Spider 是目前 SQL 生成任务中最具代表性的数据集. 近期的研究中编码模型常用预训练模型和图神经网络来增强模型对自然语言和数据库的编码能力, 并且解码模型中基于语法生成的方法往往会取得更好的效果.

在解码过程中, 也有研究针对解码过程中的通用改进方法, 这些方法可以用于多种解码模型中. Vinyals 等 [75] 提出指针网络在多种自然语言处理任务上均有应用, 指针网络允许从输入中拷贝单词到

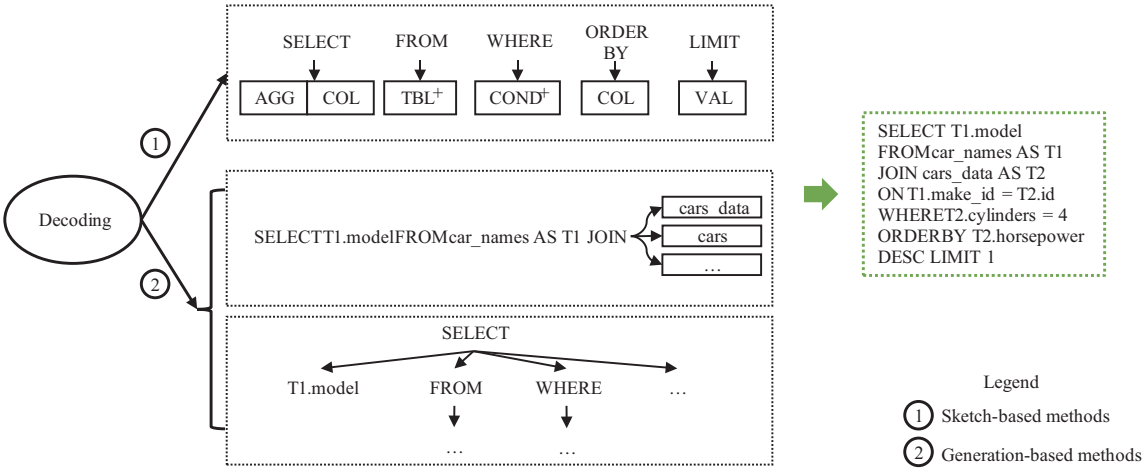

图 6 (网络版彩图) 解码过程示例

Figure 6 (Color online) An example of decoding process

表 4 经典 SQL 生成模型的技术特点

Table 4 Technical characteristics of classic SQL generation models

Ref.	Year	Model	Dataset	Encoder	Decoder
Yu et al. [50]	2018	TypeSQL	WikiSQL	Bidirectional LSTM	Simple Sketch-Based Slot-Filling
Choi et al. [72]	2020	RyanSQL	Spider	BERT	Complex Sketch-Based Slot-Filling
Hui et al. [73]	2021	SDSQL	WikiSQL	Schema Dependency Learning	Simple Sketch-Based Slot-Filling
Zhong et al. [15]	2017	Seq2SQL	WikiSQL	Bidirectional LSTM	Sequence-Based Generation
Lei et al. [60]	2020	SLSQL	Spider	BERT and Bidirectional GRU	Sequence-Based Generation
Xuan et al. [69]	2021	SeaD	WikiSQL	Pre-training Schema-aware Denoising	Sequence-Based Generation
Wang et al. [53]	2020	RATSQL	Spider	Relation-Aware Self-Attention	Grammar-Based Generation
Cao et al. [56]	2021	LGESQL	Spider	Line Graph Enhanced Encoding	Grammar-Based Generation
Rubin et al. [74]	2021	SMBOP	Spider	Relation-Aware Self-Attention	Bottom-Up Grammar-Based Generation

输出中, 这有利于模型生成词表之外的单词, 从而提升模型性能. Wang 等 [76] 提出执行指导的解码机制, 利用语义信息判断生成的部分程序的执行信息, 在解码过程中检测并排除错误程序, 从而提升模型解码效果. Kelkar 等 [77] 提出一种判别式重新排序方法 Bertrand-DR. 其将重新排序过程构建为一种基于 BERT 的分类器, 以此来判断生成的 SQL 语句是否与输入信息对应.

5.2.1 基于模版插值的方法

基于模版的 SQL 生成方法是 SQL 生成问题的一种简化形式, 仅仅生成模版中空缺位置的所需要的值. 同一种 SQL 语句可能有多种等价的表示形式, 例如查询条件中约束的顺序改变往往不影响 SQL 语句的语义和执行结果. 如果模型仅仅是因为顺序不正确而判断生成结果不正确是存在偏差的, 这种问题也被称为 “顺序重要 (order-matter)” 的问题. 模版内部每个需要填空的位置可以看作是一个集合, 并且集合内部没有顺序区分. 所以可以将模版生成方法看作序列到集合 (sequence-to-set) 的方法, 因此使用模版生成的方法有利于模型缓解顺序重要的问题.

SQL 任务中基于模版的方法在 Xu 等 [49] 的 SQLNet 网络中首先提出, 用于解决 WikiSQL 数据集中因为 SQL 语句等价形式所引起的顺序问题. SQLNet 的设计引入了序列到集合的结构, 用于预测无序的约束集, 而不是有序的序列. Yu 等 [50] 在 TypeSQL 中也使用模版插值, 并且在解码过程中将需

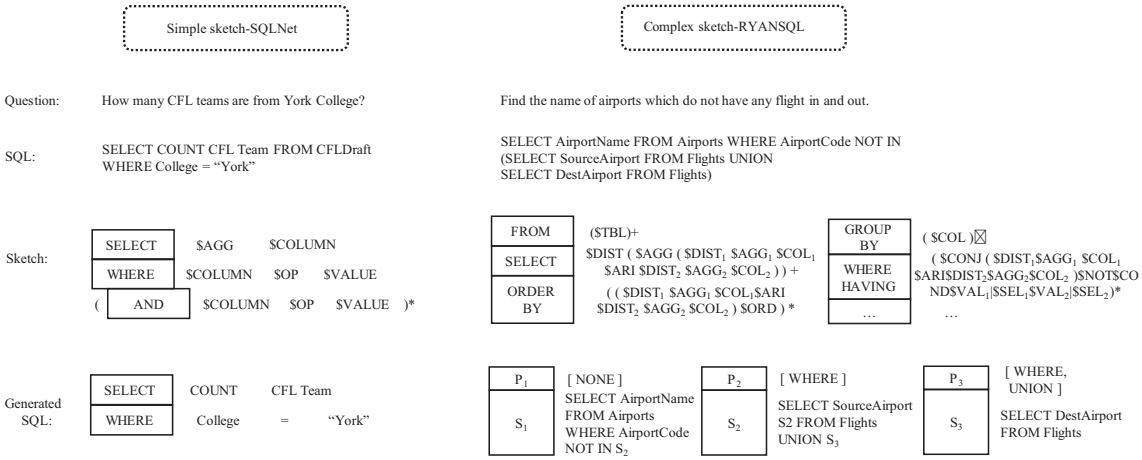

图 7 基于模版插值方法生成 SQL 示例

Figure 7 An example of generating SQL statements in sketch-based slot-filling methods

要插值的内容总结为 3 个类别并使用 3 个模型来进行训练, 以此针对不同类型的插值进行建模. Dong 和 Lapata [78] 提出 Coarse-to-Fine 模型, 将基于模版的生成任务分为两个阶段进行, 首先根据查询问题的语义生成一个粗略的模版, 忽略其中的具体的参数和变量. 然后根据自然语言输入和模版填充缺失的细节. Hwang 等 [57] 在 SQLova 中利用预训练模型的优势, 进一步提升模版中插值处的预测效果. He 等 [58] 在 SQLova 的基础上提出了 X-SQL, 将模版插值的解码过程进一步简化, 并利用上下文增强的方法提升生成效果. Lyu 等 [59] 提出 HydraNet, 更加充分地使用预训练模型的优势. 不同于 SQLova 和 X-SQL 将所有的查询问题和数据库模式输入给预训练模型, HydraNet 将自然语言和单独的列进行预训练, 通过简化解码过程中的复杂操作提升性能. Hui 等 [73] 提出 SDSQL, 使用数据库模式依赖指导的模型, 大幅度提升了解码期间的推理效率, 能够为下游任务提供更大的灵活性.

早期基于模版的方法大多针对比较简单的 SQL 语句, 如 WikiSQL 中的数据, 因为其中的 SQL 语句不包含 "GROUP BY", "JOIN" 等复杂操作. 对于复杂的 SQL 生成数据集而言, 如 Spider, 基于模版的方法将面临更多的挑战. 这些挑战主要来源于 (1) 难以定义一个通用的复杂 SQL 模版; (2) 模版中嵌套查询所需内容难以预测. Yu 等 [79] 在 Spider 数据集上首次提出了基于模版的方法 SyntaxSQLNet, 模型将 SQL 解码过程分解为 9 个模块来处理不同 SQL 组件, 每个模块中使用基于模版的方法. 不同于之前 SQLNet 使用预定义的模版, SyntaxSQLNet 使用特定的语法树来进行解析, 递归地调用不同模块来生成 SQL 语句. Lee [80] 在 Spider 数据集上改进基于模版的方法, 提出了 RCSQL. 模型中首先使用文本分类的方法给每个子句中各个模块进行分类, 然后使用特定于子句的解码器根据模版生成列和相应的运算符. 曹金超等 [81] 也针对 Spider 数据集中的复杂 SQL 语句设计了一种基于模版插值填充的方法. 针对复杂 SQL 中的多表查询问题, 该文将其 "FROM" 子句中的 "JOIN" 操作建模为斯坦纳树问题并采用全局优化算法进行求解. 最终模型在 SQL 语句相应子句上的预测准确率有一定提升. 在 Spider 数据集上最新的模版插值方法是 Choi 等 [72] 提出的 RYANSQL 模型, 其中定义了语句位置代码 (statement position code, SPC), 将嵌套的 SQL 查询转换为一组非嵌套的 SELECT 语句, 并进行递归的预测. 除此之外, RYANSQL 模型也对输入增加了处理, 如补充列名称信息, 以此方便信息的查找和区分.

图 7 展示了现有基于模版的方法可以根据 SQL 语句的复杂度分为简单模版和复杂模版两种思路.

简单模版主要面向类似于 WikiSQL 的数据集, 因为可以使用简单的模式覆盖数据集中所有的 SQL 语句. 图 7 中左子图展示了 SQLNet 中提出的简单模版的例子, 只需要设置 SELECT, WHERE 和 AND 这 3 个关键字, 模型预测聚合操作符 $AGG、相关列 $COLUMN、操作符 $OP、相应值 $VALUE 以及处理查询中可能出现的多个条件. 图 7 中右子图展示了 RYANSQL 中提出的复杂模版的例子, 相比之下, 复杂模版中需要考虑的 SQL 关键字以及关键字之间联系的复杂性明显提升. 在上述例子中, RYANSQL 将复杂的 SQL 语句根据 3 种类型的语句位置代码转化为了相应的非嵌套 SELECT 语句, 生成过程中根据每个模版递归地生成相应的内容, 并使用语句位置代码将每个模块合成为最终的 SQL 语句.

5.2.2 基于生成的方法

基于生成的方法将 SQL 语句看作一个整体进行生成, 而不是将 SQL 语句看作由一个模版中的不同模块组合而成. 与基于模版的方法相比较, 基于生成的方法在形式上更加灵活, 无需预先定义模版以及设计模版中多个模块之间的联系. 生成的方法可分为基于序列的方法和基于语法的方法, 两者的主要区分是解码过程中是否按照语法规则的约束进行生成.

基于序列生成的方法. 基于序列的 SQL 生成方法将 SQL 语句看作一种特殊的自然语言, 利用机器翻译的方法将查询问题翻译为对应的 SQL 语句. 相对于基于模版插值的方法, 基于序列的方法不用设计复杂的模版, 在形式上更加简洁, 且符合生成式方法的直观理解. 但基于序列的方法更加注重生成过程的连续性, 往往只用考虑 SQL 语句中多种表达形式的一种表达顺序.

原生的基于序列的方法往往只能作为基准模型, 并不能达到令人满意的生成效果, 所以基于序列的方法作为主要模型出现在文献中时常常会搭配其他层面的优化, 如强化学习和预训练模型等. Zhong 等 [15] 首次在大规模 SQL 生成数据集 WikiSQL 上使用序列生成的方法, 提出了 Seq2SQL 模型. 模型引入了指针网络中的拷贝机制生成 SQL 语句中的查询列名称, 使用强化学习生成 SQL 语句的条件, 缓解查询条件中由于顺序无关而无法计算交叉熵损失函数的情况. Seq2SQL 利用了数据集中 SQL 语句的特点减少搜索空间并使用强化学习中的奖励机制处理查询条件的顺序问题, 在与之前语义解析模型相比的情况下得到了较好的效果. Xuan 等 [69] 提出了一种名为 SeaD 的序列生成模型, 其中使用 Transformer 作为基础架构, 并没有增加语法约束以及其他额外模块, 以此来探索序列生成模型在应用中是否被低估. 模型中使用预训练的方法让模型学习如何在改变后的输入上预测输出以及如何在打乱后的数据中恢复输入, 并使用一种条件敏感的执行指导策略来进行解码. SeaD 在解码过程中丢弃那些无法执行或返回空结果的查询, 以及根据数据集的特点对 SQL 中不同的条件在生成过程中进行具体的处理. SeaD 通过以上做法来最大化序列生成方法的潜能, 并在 WikiSQL 数据集上取得了很好的效果.

基于语法生成的方法. 基于语法的生成方法是在序列生成的基础上增加了语法约束, 要求模型在解码过程中生成抽象语法树中的语法规则, 而不是简单的单词. 研究结果显示, 基于语法生成往往能够取得较好的性能, 尤其是面对复杂 SQL 生成的情况下, 基于语法的生成模型会明显比其他模型的性能高. 基于语法生成的方法可以分为利用语法解码和针对语法优化这两种思路.

利用语法解码的思路, 主要是利用已有的语法框架进行解码阶段的约束, 缓解基于序列的方法无法有效捕捉程序结构的问题. Rabinovich 等 [82] 在语义解析和代码生成中引入抽象语法网络 (abstract syntax networks, ASNs), 扩展了标准的编码器 – 解码器结构. 其特点是以自顶向下的方式生成抽象语法树, 在任何给定输入的解码过程中, 模块之间进行动态选择的相互递归, 其中生成的树的结构反映了递归的调用图. ASNs 使用抽象语法描述语言框架来构造 AST, 解码过程中使用特定的设计模块来

生成其中具体的语法. ASNs 在代码生成数据集和多个语义解析数据集上取得了当时最好的结果. Yin 和 Neubig [83] 提出语法神经网络模型 SNM, 使用基于语法的神经网络模型来生成通用编程语言, 如 Java, Python 等. 为了捕捉编程语言底层语法信息, 文献 [83] 首先定义模型将输入的自然语言转化为 AST, 其中 AST 由特定语言的标准解析器提供, 解码过程由一系列树的构造动作组成. 最后使用生成工具将 AST 转换为目标代码. SNM 使用结构化信息约束搜索空间, 使得生成的代码有良好的格式, 并且能够反映编程语言的特性. 2017 年, Yin 和 Neubig [84] 继续提出了一种基于转换的神经网络解析器 TRANX, 使用生成目标的语法信息约束输出空间以及对信息流建模. TRANX 模型中使用 ASDL 中指定的 AST 作为通用的中间语义表示, 并将任务特定的语法作为外部知识提供给模型作为指导信息, 因此将语义解析过程与特定语法解耦. TRANX 模型最终在 WikiSQL 和多个语义解析数据集的原有基础上提升了性能. Zhao 等 [85] 提出了一种关于解码过程中语法的预训练模型 GP, 用于在解码端减少解码过程中出现的语法错误. 由于 SQL 语法与具体的自然语言无关, 所以 GP 首先在没有编码器信息时预训练了一个解码器, 并使用 flooding 作为损失函数, 用于避免局部最优的情况. GP 有效地提高了模型的训练效率, 具有良好的鲁棒性和收敛性, 并在 Spider 数据集上具有较高的准确率. Xu 等 [86] 提出 DT-Fixup 模型, 证明通过适当的初始化和优化, 深层 Transformer 的好处可以延续到具有小规模数据集的挑战性任务上. DT-Fixup 模型训练了 48 层的 Transformer, 包括了 RoBERTa 的 24 层微调层和 24 个从头训练关系感知层, 最终使用基于语法的生成方法进行解码. 实验结果表明对于需要推理和结构理解的困难任务, 如 SQL 生成, 增加深度有助于提高在小数据集上的泛化能力, 并在 Spider 数据集上达到接近最好的效果.

针对语法优化的思路中, 研究者主要针对原生的语法建模方法进行优化, 旨在设计更加有效的语法表示以及更有效地利用语法信息. Lin 等 [87] 提出 GrammarSQL, 其为 SQL 语句设计了一种语法, 它能够覆盖 ATIS 和 Spider 中 98% 的语法情况. 设计的新语法考虑了在生成过程中语法的依赖情况, 如数据库模式中的列名称和值之间的约束关系. 首先, 对一些上下文敏感度的情况通过向上下文无关语法添加额外的非终结符来处理, 并以此来确保表、列和值引用的一致性. 然后对于更复杂的上下文敏感度, 例如确保 SQL 查询中的表连接共享公共外键, 则对产生式规则扩展使用运行时约束以确保只允许有效的程序. 这种新的语法约束解码让 GrammarSQL 模型在 ATIS 和 Spider 上达到了相对较好的结果. Guo 等 [88] 提出 IRNET, 与之前端到端的方法不同, IRNET 将 SQL 分为 3 个步骤进行生成, 首先在编码部分进行数据库模式链接, 然后生成 SQL 语句的中间表示 (intermediate representation, IR), 最后根据中间表示生成最终的 SQL 语句. IRNET 设计了一种特定领域的语言, SemQL, 来作为自然语言和 SQL 语句的中间表示. 以此来隐藏特定的 SQL 语法实现细节, 缓解自然语言和 SQL 语句之间的不匹配问题, 从而充分利用自然语言表达的信息. 解码器利用基于语法的方法, 通过一系列构造树结构的活动建模 SemSQL 的生成过程. 模型最终在复杂跨域数据集 Spider 上取得了较好的效果. Rubin 等 [74] 注意到在基于语法的方法中, 原始自顶向下的解析方法效率不高且常常解析出无意义的部分树, 所以提出 SmBoP, 选择使用自底向上的方法生成语法树. 其中在语法层面也借鉴了 IRNET 的思路, 使用增强后的关系代数作为生成 SQL 的中间表示形式. 具体而言, 在第 t 步解码过程中, 解码器会并行地生成前 K 个子树, 其中每个子树的高度小于等于 t. 这样可以有效地降低运行时的复杂程度, 从而提升效率. 在建模方面, 自下而上的解码方式有利于构造有意义的子程序, 而不是没有清晰语义的部分代码. 利用这样的方法使得 SmBoP 能够在 Spider 数据集上达到目前最好的效果, 以及提升模型解码效率.

图 8 展示了基于语法的两种思路. 左子图展示了利用 SQL 通用语法在解码阶段进行约束的例子, 一系列研究证明增加语法约束会提升模型的生成效果. 右子图展示了针对现有语法进行优化的思路,

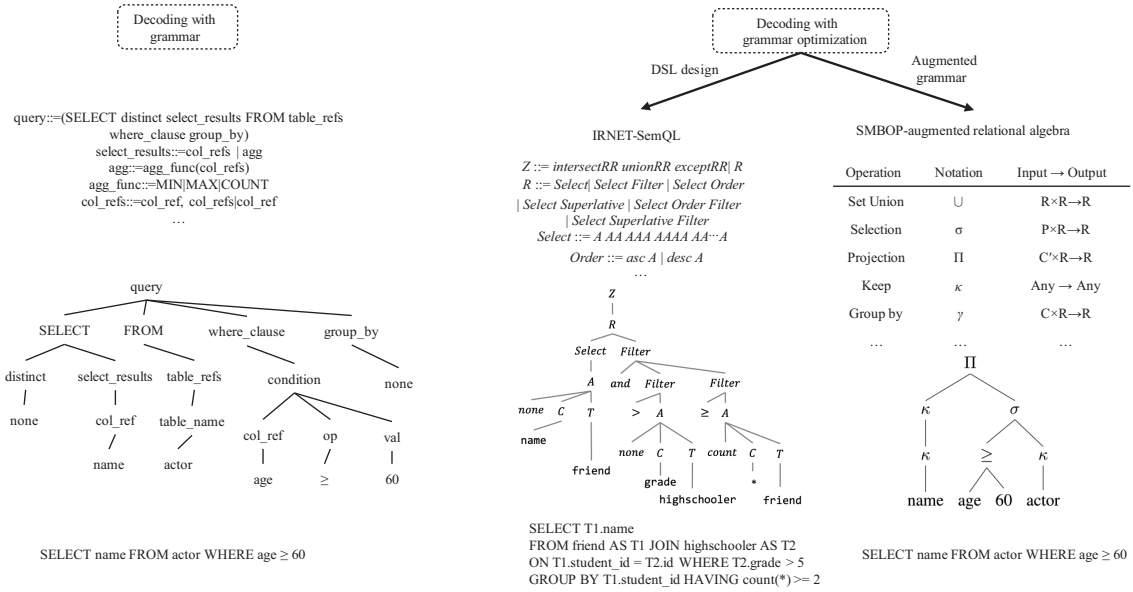

图 8　基于语法的方法生成 SQL 示例

Figure 8　An example of generating SQL statements in grammar-based methods

包括了特定领域语言 (domain specific language, DSL) 的设计和增强现有语法在 SQL 生成中的表示能力两种具体做法, 这种思路旨在弥补自然语言到 SQL 语法之间存在的差距, 通过中间表示的进一步增强将输入转化为 SQL 语句的能力. IRNET 通过设计中间表示的语法提升 SQL 生成效果, 图中展示的语法树例子中, SQL 查询中的 “GROUP BY”, “HAVING” 和 “FROM” 子句在 SemQL 查询中被消除, “WHERE” 和 “HAVING” 中的条件在 SemQL 查询的 Filter 子树中被统一表示, 以此来弥补自然语言和 SQL 实现细节之间的差距. 实现细节可以在随后的推断阶段通过领域知识从 SemQL 查询中确定. SMBOP 模型利用 SQL 特定的操作增强现有的关系代数 (relational algebra) 表示形式, 并使用自底向上的方式生成子程序, 以此来进一步提升模型基于语法生成的性能. 图中 SBMOP 对应的关系代数中, R 表示关系, P 表示断言, C 表示常量, C' 表示常量集合. 在构造树的过程中还引入了了一元 KEEP 操作, 它不会改变应用它的子树的语义. 因此, 构造过程中以在不改变形式查询的情况下增加树的高度, 以此保证生成子树为平衡树.

5.3　小结

随着深度学习的发展, 研究者不断地探索和构造各种模型结构来适配和解决不同领域面临的问题. 在 SQL 生成领域也是如此, 得益于深度学习和自然语言处理领域的快速发展, 大量优秀的模型结构和建模思路被用于生成 SQL 语句. 同时, 研究者们不断地优化和改进现有模型, 使其能更好地解决 SQL 生成面临的问题, 这也使得 SQL 生成的模型结构可以被其他相似领域所借鉴并促进深度学习的发展. 在 SQL 生成的模型结构研究工作中, 研究问题主要涉及对结构化数据库编码问题、自然语言和数据库之间模式链接问题以及生成具有严格语法约束的 SQL 语句问题. 针对这些主要问题, 一系列研究工作中引入图神经网络和预训练模型等先进技术来优化解决方案. 研究结果证明针对 SQL 特点的模型结构创新, 如针对 SQL 任务的预训练模型, 能够有效地提升实验性能. 后续的研究应继续针对 SQL 生成所涉及的主要问题进行模型结构方面的创新, 继续探索深度学习模型如何增强对自然语言和数据库之间语义匹配和链接的捕获能力, 以及对不同类型解码方式生成 SQL 的潜力.

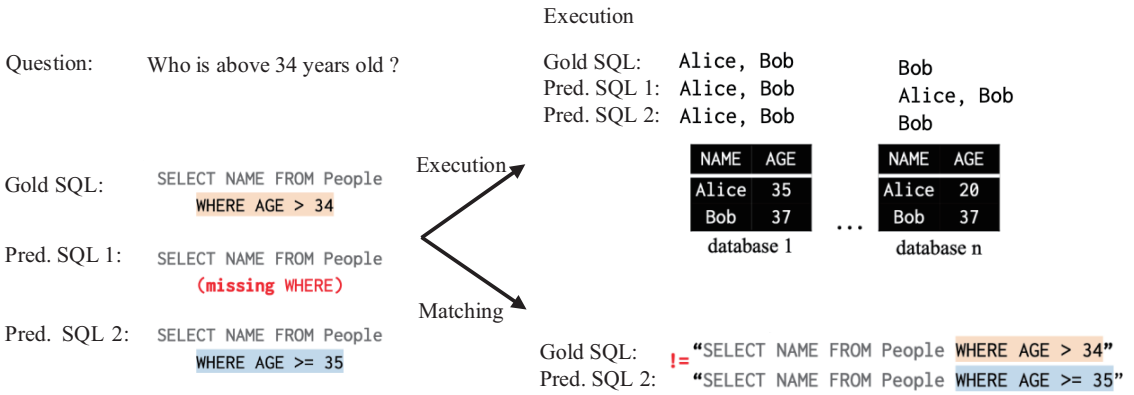

图 9 (网络版彩图) 执行和完全匹配指标评估 SQL 生成效果可能存在不足的示例

Figure 9 (Color online) Examples of possible inadequacies in evaluating the generated SQL with execution accuracy and exact matching

6 SQL 生成评估方法

评估方法用于评估深度学习模型生成 SQL 的效果, 针对不同的生成方式可以分为单轮 SQL 生成的评估和多轮 SQL 生成的评估两类. 单轮 SQL 生成的评估方法包括了基于文本和基于执行两种. 多轮 SQL 生成的评估方法参考自然语言多轮对话的评估指标, 包括了查询匹配 (question match) 和交互匹配 (interaction match).

单轮评估指标中, 基于文本的评价指标只需利用生成的文本进行计算, 包括组成成分匹配率 (component matching) 和完全匹配率 (exact matching). SQL 组成成分匹配率是关于 SQL 内部实现细节对应的匹配率统计指标. 计算过程首先需要将完整的 SQL 拆分为各个部分, 如 “SELECT”, “WHERE” 和 “GROUP BY” 等部分, 接着计算每个部分中应当包含的值的集合是否匹配. 每个不同部分的匹配率可以进一步计算相应的 F1 值, 用于权衡精准率和召回率并得到更客观的评价. 完全匹配率是指生成预测的 SQL 语句和正确的 SQL 语句完全相同的概率. 与 SQL 组成成分匹配相比较, 完全匹配的要求更高, 即一个 SQL 语句中所有组成成分都匹配后才能算作该 SQL 语句属于完全匹配. 值得注意的是 SQL 语句某些成分中所包含的值没有顺序的限制, 所以在无顺序要求的成分中需要以集合的形式来计算是否匹配, 而不是以字符串的形式进行计算, 以此来消除仅仅因为顺序不匹配所带来的误差. 基于执行的评价指标需要对比 SQL 语句执行后的结果, 执行结果正确率 (execution accuracy) 是统计执行结果匹配程度的指标. 由于同一种功能的 SQL 语句也可能存在多种的表达形式, 为了避免完全匹配指标中遗漏的情况, 查看 SQL 语句的执行结果来判断生成的 SQL 是否正确也是一个重要的指标. 使用执行正确率作为评价指标要求数据集中存在可以进行单元测试的条件, 即要求数据集中提供正确 SQL 语句所生成的查询结果, 并以此来与生成 SQL 的执行结果进行对比.

多轮评估指标中, 查询匹配是指在仅考虑单轮生成的情况下整体的匹配率, 一般使用单轮评估指标中的完全匹配率作为相应指标. 在实际实验中, 也需要针对当前单轮生成过程中是否依赖前文生成结果进行分开讨论. 交互匹配是指在整体多轮生成过程中, 内部的所有查询问题和生成结果是否正确.

然而, 单纯的基于文本匹配和 SQL 执行的评估指标往往存在不足. 如图 9 所示, 预测的 SQL 1 相比于正确的 SQL 缺少了查询条件, 预测的 SQL 2 与正确的 SQL 不完全匹配. 但只用 database 1 作为测试的情况下, 预测的 SQL 1 的执行结果仍然正确, 所以为假阳性样例, 即执行结果正确但实际 SQL

存在语义错误. 而 SQL 2 虽然不完全匹配, 但在语义上与正确的 SQL 相同 (在数据库中年龄均用整数表示的情况下, > 34 和 $\geqslant 35$ 的查询结果相同), 所以 SQL 2 是一个假阴性样例, 即 SQL 预测正确但是被评估指标分到了错误那一类.

目前也有研究关注如何优化 SQL 生成的评估方法, Zhong 等 [89] 提出了蒸馏测试套件的方法, 使用测试套件正确率来评估模型生成 SQL 的效果. 首先通过修改数据集中正确的 SQL 语句得到一系列形式上相似的 "邻居查询", 如图 9 所示, 预测 SQL 1 相比较于正确 SQL 少了条件, 算是一种邻居查询. 然后随机生成一系列的数据库, 如图 9 中的 database 1 $\sim$ database n. 最后通过蒸馏的方式保留能区分正确查询 SQL 和邻居查询 SQL 语义的最小数据库数量作为测试套件, 用于评估生成 SQL 的质量. 实验使用 21 个 Spider 数据集榜单上模型所预测的 100 个随机样本, 实验结果表明完全匹配率存在低估模型性能的情况, 平均导致了 2.5% 的假阴性, 最高达到 8.1% 的假阴性情况. 并且在复杂的 SQL 语句上, 完全匹配率更加容易出错. 这也进一步说明单纯使用完全匹配率无法客观反应 SQL 生成模型的性能, 后续关于生成 SQL 质量评估的研究仍然需要研究者的关注.

7 总结与展望

本文综述了现有基于深度学习的 SQL 生成相关工作, 从 SQL 生成场景、数据集、模型结构和评估方法 4 个层面对现有工作进行分类. 在此分类的基础上, 本文又针对每个类别进行详细的划分, 并总结整理了关于每个研究方向的相关工作和研究思路. SQL 生成能够让非专业人士轻松地访问数据库内容, 而无需了解复杂的 SQL 语法, 降低了从大规模数据中抽取有效信息的门槛. SQL 自动生成是编程自动化和数据驱动的软件工程中重要的研究课题, 具有重要的学术意义和广泛的应用价值.

基于深度学习的 SQL 生成已经有了大量的研究成果, 但仍然存在如下的挑战.

(1) 数据集方面, 缺乏面向实际应用场景的高质量数据. 数据集是 SQL 生成任务中的重要研究方向, 越来越多研究关注 SQL 生成数据集的构造, 其中数据的规模和质量是决定数据集是否可用的基础条件, 数据集的领域覆盖程度、查询问题表达和意图多样性对构造一个好的数据集至关重要. 除此之外, 使用特定数据集训练的模型能否在实际场景中应用也是数据集构造的关键挑战, 这不仅要求数据集具有良好的规模和质量, 还要求数据符合应用的需求. 近期的研究中, 研究者提出了一些上下文依赖的跨域数据集, 模拟了实际应用中与 SQL 生成系统对话的形式, 考虑了当前 SQL 语句的生成依赖前文生成结果的情况. 但目前 SQL 生成仍难以在真实场景中部署, 实际应用场景中的 SQL 生成数据集仍然值得继续探索.

(2) 编码模型方面, 处理数据库相关信息的编码能力不足. 数据库编码是 SQL 生成任务的重要特点, 目前关于数据库编码的研究主要关注数据库模式, 部分研究中数据库记录也被用来提升整体编码效果. 数据库模式中模式编码和模式链接最具有挑战性, 目前的研究大多基于注意力机制、图网络和预训练模型. 但在图网络中, 现有的研究主要为构建以节点为中心的图, 对于节点局部与非局部的关系, 以及边的拓扑结构处理能力不足. 在预训练模型中, 现有的大多研究主要使用已有的预训练模型进行微调, 如 BERT, 较少有研究针对 SQL 生成任务的特点进行预训练框架的优化, 导致现有的研究没有充分利用预训练模型的潜力. 对于使用数据库记录增强编码的情况, 如何既充分利用数据库记录又能让数据库避免过多暴露数据记录的方法同样值得研究. 一个合适的数据库编码有利于充分地捕捉查询问题和数据库信息, 也能够对后续的解码结果具有积极的影响, 图网络、预训练模型和数据库记录增强的方法仍然需要研究者进一步探索.

(3) 解码模型方面, 缺乏对不同类型解码方式的潜力进行深入探索. 目前在解码过程中主要包含

基于模版、基于序列和基于语法的 3 种具体方法. 在基于模版的方法中, 针对 SQL 语句不同复杂程度的数据集存在不同的模版. 在复杂模版中不同的设计方法也会对模型的解码性能产生重要的影响, 需要对模版进行合适设计以及对模版内部生成方法进行合适的选择才能进一步提升模版的效果. 基于序列的生成方法是生成方式中的基础方法, 但也有近期研究表明基于序列生成的潜力没有被充分探索. 目前大多 SQL 生成方法采用基于语法的方式, 但在基于语法生成方式内部, 也存在不同的改进和优化策略, 如新的表示方法设计和自下向上解码方式等. 基于语法生成的方式是目前最具潜力的方法, 基于该方式的优化往往能够提升生成效果. 在各种解码方式内部大多使用注意力机制, 指针网络 [75] 和执行指导 [76] 的解码等机制, 在不同解码方法下的通用优化策略依然存在进一步探索的空间. 总之, 在解码过程中, 针对不同解码方式的内部优化, 以及通用的解码指导性方法仍然需要研究者进一步的探索.

(4) 评估模型的指标需要进一步完善. 目前评估生成 SQL 质量的指标主要使用文本匹配程度以及执行结果的正确率, 但这些评价指标依然面临如下问题: 相同语义的 SQL 语句可能有不同的表现形式; 对于执行结果为空时, 执行结果是否准确难以判断; 语义不同的 SQL 语句可能在特定数据库上有相同的执行结果; 对于不了解 SQL 的人员来说, 难以在不检查数据库的情况下判断生成的结果是否真正符合事实; 针对不同数据集训练的模型, 难以评价不同模型泛化的能力. 虽然现有研究中大多使用多个指标进行综合评估, 但用于解决上述问题的新评估指标仍然有待进一步探索.

为了应对如上的一系列挑战, 下面讨论未来基于深度学习的 SQL 生成研究可能的方向.

(1) 进一步挖掘真实场景中自动生成 SQL 的需求, 以真实场景中的具体需求为导向, 构造高质量高可用性的真实数据集. SQL 生成数据集是支撑基于深度学习的 SQL 生成系统的基础, 不同的数据决定了 SQL 生成任务的特点. 未来研究可以进一步挖掘真实场景中 SQL 自动生成的需求, 如查询问题使用多种语言描述的情况、SQL 生成错误后需要修复情况和 SQL 语句外依赖其他编程语言程序执行的情况. 对真实场景中数据的挖掘是 SQL 生成研究中非常重要的一步, 真实数据库中的数据库模式和数据记录往往与自然语言存在较大的分布差异. 这将要求模型具有更强的语义表示能力和组合泛化能力, 同时, 这样的模型对于工业界具体应用场景的落地更有价值.

(2) 降低大规模数据集标注的开销. 现有研究中大规模数据集对 SQL 生成的研究有极大的促进作用, 但这些数据集往往也会消耗大量的人力物力进行标注. Spider 数据集 [17] 的构造过程要求标注者在给定数据库的情况下给出查询问题和相应的 SQL 语句. DuSQL [21] 数据集在构造过程中首先根据自定义的语法生成 SQL 语句并且要求标注者将伪查询问题改为真实的自然语言描述问题, 以此降低标注的成本. 降低数据的标注开销不仅能够帮助研究人员快速构造符合需求场景的数据集, 也能为训练模型提供更多的数据, 从而提升模型的效果, 未来在数据集构造的研究方面应同时考虑数据质量和降低数据开销两方面.

(3) 构建 SQL 生成的相关深度学习模型需要针对 SQL 生成的特点进行进一步的研究. 编码模型是 SQL 生成的关键, 其中不仅需要特别地考虑所涉及的数据库, 也需要考虑自然语言和数据库的关系. 并且编码模型决定了模型能否充分利用自然语言的查询问题和结构化的数据库信息. 在未来编码模型的研究中需要充分利用现有深度学习的最新研究成果, 如图神经网络和预训练模型等, 并针对 SQL 生成任务的特点进行优化. 尤其是针对预训练模型, 应当继续探索 SQL 生成任务特定的高效预训练方法, 并充分地利用数据库中数据记录的内容, 进一步提升 SQL 生成效果. 在未来解码阶段, 针对不同方法的内部优化和通用解码优化方案依然需要创新, 进一步提升 SQL 生成的性能. 并且要注重编码模型和解码模型的综合考虑, 这样才能充分发挥每一种解码方式的潜力. 但模型的创新并不一定指的是增加模型的复杂度, 尤其是解码阶段, 未来应当简化模型结构并提升模型效率, 这样才能在工业界提供

更好的解决方案.

(4) 评估 SQL 生成效果的指标和方法的探索. 评估方法用于衡量深度学习模型生成 SQL 语句的质量, 一个好的评估方法应尽量避免假阳性和假阴性样本的出现. 已有研究证明评估方法中不能单纯使用完全匹配和执行的指标 [89], 应当充分考虑 SQL 的语义, 避免具有正确语义而形式不同的 SQL 语句被误报. 在语义层次之外, 评估指标也需要考虑生成 SQL 语句在时间空间上的执行效率. 最后, 由于目前 SQL 生成面向非专业的用户, 所以评估指标也应该考虑什么样的生成结果是用户可以接受的, 并且尽量保证执行 SQL 后输出结果的可读性.

参考文献

1 ISO/IEC. Information technology—Database languages—SQL—Part 1: Framework (SQL/Framework). 9075-1:2016. https://www.iso.org/standard/45498.html

2 Popescu A M, Etzioni O, Kautz H. Towards a theory of natural language interfaces to databases. In: Proceedings of the 8th International Conference on Intelligent User Interfaces, 2003. 149–157

3 Yaghmazadeh N, Wang Y, Dillig I, et al. SQLizer: query synthesis from natural language. In: Proceedings of the ACM on Programming Languages, 2017. 1–26

4 Kalajdjieski J, Toshevska M, Stojanovska F. Recent advances in SQL query generation: a survey. 2020. ArXiv:2005.07667

5 Katsogiannis-Meimarakis G, Koutrika G. A deep dive into deep learning approaches for text-to-SQL systems. In: Proceedings of the 2021 International Conference on Management of Data, 2021. 2846–2851

6 Wang D C, Appel A W, Korn J L, et al. The zephyr abstract syntax description language. In: Proceedings of the Conference on Domain-Specific Languages, 1997. 1–15

7 Pan X, Xu S H, Cai X R, et al. Survey on deep learning based natural language interface to database. J Comput Res Develop, 2021, 58: 1925–1950 [潘璇, 徐思涵, 蔡祥睿, 等. 基于深度学习的数据库自然语言接口综述. 计算机研究与发展, 2021, 58: 1925–1950]

8 Price P. Evaluation of spoken language systems: the ATIS domain. In: Proceedings of the Workshop on Speech and Natural Language, 1990. 91–95

9 Iyer S, Konstas I, Cheung A, et al. Learning a neural semantic parser from user feedback. 2017. ArXiv:1704.08760

10 Zelle J M, Mooney R J. Learning to parse database queries using inductive logic programming. In: Proceedings of the National Conference on Artificial Intelligence, 1996. 1050–1055

11 Tang L R, Mooney R. Automated construction of database interfaces: intergrating statistical and relational learning for semantic parsing. In: Proceedings of Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 2000. 133–141

12 Li F, Jagadish H V. Constructing an interactive natural language interface for relational databases. Proc VLDB Endow, 2014, 8: 73–84

13 Finegan-Dollak C, Kummerfeld J K, Zhang L, et al. Improving text-to-SQL evaluation methodology. 2018. ArXiv:1806.09029

14 Dahl D A, Bates M, Brown M K, et al. Expanding the scope of the ATIS task: the ATIS-3 corpus. In: Proceedings of the Workshop on Human Language Technology, 1994. 43–48

15 Zhong V, Xiong C, Socher R. Seq2SQL: generating structured queries from natural language using reinforcement learning. 2017. ArXiv:1709.00103

16 Shi T, Zhao C, Boyd-Graber J, et al. On the potential of lexico-logical alignments for semantic parsing to SQL queries. 2020. ArXiv:2010.11246

17 Yu T, Zhang R, Yang K, et al. Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. 2018. ArXiv:1809.08887

18 Lee C H, Polozov O, Richardson M. KaggleDBQA: realistic evaluation of text-to-SQL parsers. 2021. ArXiv:2106.11455

19 Min Q, Shi Y, Zhang Y. A pilot study for chinese SQL semantic parsing. 2019. ArXiv:1909.13293

20 Sun N, Yang X, Liu Y. TableQA: a large-scale chinese text-to-SQL dataset for table-aware SQL generation. 2020. ArXiv:2006.06434

21 Wang L, Zhang A, Wu K, et al. ChiTeSQL: a large-scale and pragmatic Chinese text-to-SQL dataset. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020. 6923–6935

22 Yu T, Zhang R, Yasunaga M, et al. SParC: cross-domain semantic parsing in context. 2019. ArXiv:1906.02285

23 Yu T, Zhang R, Er H Y, et al. CoSQL: a conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. 2019. ArXiv:1909.05378

24 Guo J, Si Z, Wang Y, et al. CHASE: a large-scale and pragmatic chinese dataset for cross-database context-dependent text-to-SQL. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, 2021. 2316–2331

25 Pasupat P, Liang P. Compositional semantic parsing on semi-structured tables. 2015. ArXiv:1508.00305

26 Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model. J Mach Learn Res, 2003, 3: 1137–1155

27 Mikolov T, Karafiát M, Burget L, et al. Recurrent neural network based language model. In: Proceedings of the 11th Annual Conference of the International Speech Communication Association, 2010

28 Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. In: Proceedings of Advances in Neural Information Processing Systems, 2013. 3111–3119

29 Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space. 2013. ArXiv:1301.3781

30 Pennington J, Socher R, Manning C D. Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014. 1532–1543

31 Joulin A, Grave E, Bojanowski P, et al. Bag of tricks for efficient text classification. 2016. ArXiv:1607.01759

32 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of Advances in Neural Information Processing Systems, 2017. 5998–6008

33 Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9: 1735–1780

34 Sundermeyer M, Schlüter R, Ney H. LSTM neural networks for language modeling. In: Proceedings of the 13th Annual Conference of the International Speech Communication Association, 2012

35 Cho K, van Merriënboer B, Bahdanau D, et al. On the properties of neural machine translation: encoder-decoder approaches. 2014. ArXiv:1409.1259

36 Li G J, Liu H, Li G, et al. LSTM-based argument recommendation for non-API methods. Sci China Inf Sci, 2020, 63: 190101

37 Tao C Q, Bao P P, Huang Z Q. Code line generation based on deep context-awareness of onsite programming. Sci China Inf Sci, 2020, 63: 190106

38 Sordoni A, Bengio Y, Vahabi H, et al. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, 2015. 553–562

39 Zhang H, Lan Y, Pang L, et al. ReCoSa: detecting the relevant contexts with self-attention for multi-turn dialogue generation. 2019. ArXiv:1907.05339

40 Peters M E, Neumann M, Iyyer M, et al. Deep contextualized word representations. 2018. ArXiv:1802.05365

41 Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training. OpenAI Blog, 2018. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

42 Devlin J, Chang M W, Lee K, et al. BERT: pre-training of deep bidirectional transformers for language understanding. 2018. ArXiv:1810.04805

43 Lewis M, Liu Y, Goyal N, et al. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. 2019. ArXiv:1910.13461

44 Cai Y, Wan X. IGSQL: database schema interaction graph based neural model for context-dependent text-to-SQL generation. 2020. ArXiv:2011.05744

45 Radford A, Wu J, Child R, et al. Language models are unsupervised multitask learners. OpenAI blog, 2019, 1: 9

46 Brown T B, Mann B, Ryder N, et al. Language models are few-shot learners. 2020. ArXiv:2005.14165

47 Liu Y, Ott M, Goyal N, et al. RoBERTa: a robustly optimized BERT pretraining approach. 2019. ArXiv:1907.11692

48 Feng Z, Guo D, Tang D, et al. CodeBERT: a pre-trained model for programming and natural languages. 2020. ArXiv:2002.08155

49 Xu X, Liu C, Song D. SQLNet: generating structured queries from natural language without reinforcement learning.

2017. ArXiv:1711.04436
50 Yu T, Li Z, Zhang Z, et al. TypeSQL: knowledge-based type-aware neural text-to-SQL generation. 2018. ArXiv:1804.09769
51 Bogin B, Gardner M, Berant J. Representing schema structure with graph neural networks for text-to-SQL parsing. 2019. ArXiv:1905.06241
52 Bogin B, Gardner M, Berant J. Global reasoning over database structures for text-to-SQL parsing. 2019. ArXiv:1908.11214
53 Wang B, Shin R, Liu X, et al. RAT-SQL: relation-aware schema encoding and linking for text-to-SQL parsers. 2019. ArXiv:1911.04942
54 Scholak T, Li R, Bahdanau D, et al. DuoRAT: towards simpler text-to-SQL models. 2020. ArXiv:2010.11119
55 Chen Z, Chen L, Zhao Y, et al. ShadowGNN: graph projection neural network for text-to-SQL parser. 2021. ArXiv:2104.04689
56 Cao R, Chen L, Chen Z, et al. LGESQL: line graph enhanced text-to-SQL model with mixed local and non-local relations. 2021. ArXiv:2106.01093
57 Hwang W, Yim J, Park S, et al. A comprehensive exploration on wikisql with table-aware word contextualization. 2019. ArXiv:1902.01069
58 He P, Mao Y, Chakrabarti K, et al. X-SQL: reinforce schema representation with context. 2019. ArXiv:1908.08113
59 Lyu Q, Chakrabarti K, Hathi S, et al. Hybrid ranking network for text-to-SQL. 2020. ArXiv:2008.04759
60 Lei W, Wang W, Ma Z, et al. Re-examining the role of schema linking in text-to-SQL. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020. 6943–6954
61 Zhang R, Yu T, Er H Y, et al. Editing-based SQL query generation for cross-domain context-dependent questions. 2019. ArXiv:1909.00786
62 Lin X V, Socher R, Xiong C. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. 2020. ArXiv:2012.12627
63 Ma J, Yan Z, Pang S, et al. Mention extraction and linking for SQL query generation. 2020. ArXiv:2012.10074
64 Zhong V, Lewis M, Wang S I, et al. Grounded adaptation for zero-shot executable semantic parsing. 2020. ArXiv:2009.07396
65 Yu T, Wu C S, Lin X V, et al. GraPPa: grammar-augmented pre-training for table semantic parsing. 2020. ArXiv:2009.13845
66 Yin P, Neubig G, Yih W, et al. TaBERT: pretraining for joint understanding of textual and tabular data. 2020. ArXiv:2005.08314
67 Lehmberg O, Ritze D, Meusel R, et al. A large public corpus of web tables containing time and context metadata. In: Proceedings of the 25th International Conference Companion on World Wide Web, 2016. 75–76
68 Shi P, Ng P, Wang Z, et al. Learning contextual representations for semantic parsing with generation-augmented pre-training. 2020. ArXiv:2012.10309
69 Xuan K, Wang Y, Wang Y, et al. SeaD: end-to-end text-to-SQL generation with schema-aware denoising. 2021. ArXiv:2105.07911
70 Liu Q, Yang D, Zhang J, et al. Awakening latent grounding from pretrained language models for semantic parsing. 2021. ArXiv:2109.10540
71 Deng X, Awadallah A H, Meek C, et al. Structure-grounded pretraining for text-to-SQL. 2020. ArXiv:2010.12773
72 Choi D H, Shin M C, Kim E G, et al. RYANSQL: recursively applying sketch-based slot fillings for complex text-to-SQL in cross-domain databases. Comput Linguist, 2021, 47: 309–332
73 Hui B, Shi X, Geng R, et al. Improving text-to-SQL with schema dependency learning. 2021. ArXiv:2103.04399
74 Rubin O, Berant J. SmBoP: semi-autoregressive bottom-up semantic parsing. 2020. ArXiv:2010.12412
75 Vinyals O, Fortunato M, Jaitly N. Pointer networks. 2015. ArXiv:1506.03134
76 Wang C, Tatwawadi K, Brockschmidt M, et al. Robust text-to-SQL generation with execution-guided decoding. 2018. ArXiv:1807.03100
77 Kelkar A, Relan R, Bhardwaj V, et al. Bertrand-DR: improving text-to-SQL using a discriminative re-ranker. 2020. ArXiv:2002.00557
78 Dong L, Lapata M. Coarse-to-fine decoding for neural semantic parsing. 2018. ArXiv:1805.04793

79 Yu T, Yasunaga M, Yang K, et al. SyntaxSQLNet: syntax tree networks for complex and cross-domaintext-to-SQL task. 2018. ArXiv:1810.05237
80 Lee D. Clause-wise and recursive decoding for complex and cross-domain text-to-SQL generation. 2019. ArXiv:1904.08835
81 Cao J C, Huang T, Chen G, et al. Research on technology of generating multi-table SQL query statement by natural language. J Front Comput Sci Technol, 2020, 14: 1133–1141 [曹金超, 黄滔, 陈刚, 等. 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: 1133–1141]
82 Rabinovich M, Stern M, Klein D. Abstract syntax networks for code generation and semantic parsing. 2017. ArXiv:1704.07535
83 Yin P, Neubig G. A syntactic neural model for general-purpose code generation. 2017. ArXiv:1704.01696
84 Yin P, Neubig G. TRANX: a transition-based neural abstract syntax parser for semantic parsing and code generation. 2018. ArXiv:1810.02720
85 Zhao L, Cao H, Zhao Y. GP: context-free grammar pre-training for text-to-SQL parsers. 2021. ArXiv:2101.09901
86 Xu P, Kumar D, Yang W, et al. Optimizing deeper transformers on small datasets. 2021. ArXiv:2012.15355
87 Lin K, Bogin B, Neumann M, et al. Grammar-based neural text-to-SQL generation. 2019. ArXiv:1905.13326
88 Guo J, Zhan Z, Gao Y, et al. Towards complex text-to-SQL in cross-domain database with intermediate representation. 2019. ArXiv:1905.08205
89 Zhong R, Yu T, Klein D. Semantic evaluation for text-to-SQL with distilled test suites. 2020. ArXiv:2010.02840

A survey of deep learning based text-to-SQL generation

Qingyuan LIANG[1], Qihao ZHU[2], Zeyu SUN[2], Lu ZHANG[2*], Wenjie ZHANG[2], Yingfei XIONG[2], Guangtai LIANG[3] & Lian YU[1]

1. *School of Software & Microelectronics, Peking University, Beijing* 102600*, China*;
2. *Key Lab of High Confidence Software Technologies (PKU), Ministry of Education, Beijing* 100871*, China*;
3. *Software Analysis Lab, Huawei Cloud, Beijing* 100095*, China*
* Corresponding author. E-mail: zhanglu@sei.pku.edu.cn

Abstract Text-to-SQL is an important application of automation software engineering. It is also a research hotspot in the field of semantic parsing. The text-to-SQL task aims to automatically generate the SQL statement according to the natural language description. It allows nonprofessionals to access the database without understanding SQL syntax. With the development of large-scale text-to-SQL datasets and artificial intelligence technologies, the text-to-SQL task is also making great progress. Compared with the traditional text-to-SQL generation, the deep learning-based text-to-SQL has the advantages of high accuracy, flexibility, and iterative learning. In recent years, several studies have focused on SQL generation based on deep learning. This research summarizes existing works from the aspects of text-to-SQL scenarios, datasets, model structures, and evaluation methods.

Keywords text-to-SQL, semantic parsing, deep learning, code generation, encoder-decoder