一种利用词典扩展数据库模式信息的 Text2SQL 方法

于晓昕，何　东，叶子铭，陈　黎，于中华
(四川大学计算机学院，成都 610065)

摘　要：现有 Text2SQL 方法严重依赖表名和列名在自然语言查询中的显式提及，在同物异名的实际应用场景中准确率急剧下降. 此外，这些方法仅仅依赖数据库模式捕捉数据库建模的领域知识，而数据库模式作为结构化的元数据，其表达领域知识的能力是非常有限的，即使有经验的程序员也很难仅从数据库模式完全领会该数据库建模的领域知识，因此程序员必须依赖详细的数据库设计文档才能构造 SQL 语句以正确地表达特定的查询. 为此，本文提出一种利用词典扩展数据库模式信息的 Text2SQL 方法，该方法从数据库表名和列名解析出其中的单词或短语，查询词典获取这些单词或短语的语义解释，将这些解释看成是相应表名或列名的扩展内容，与表名、列名及其他数据库模式信息(主键、外键等)相结合，作为模型的输入，从而使模型能够更全面地学习数据库建模的应用领域知识. 在 Spider-syn 和 Spider 数据集上进行的实验说明了所提出方法的有效性，即使自然语言查询中使用的表名和列名与数据库模式中对应的表名和列名完全不同，本文方法也能够得到较好的 SQL 翻译结果，明显优于最新提出的抗同义词替换攻击的方法.

关键词：数据库模式；语义扩展；解释信息；Text2SQL
中图分类号：TP391　　**文献标志码**：A　　DOI：10.19907/j.0490-6756.2024.012004

A Text2SQL method utilizing database schema information expanded by dictionary

YU Xiao-Xin, *HE Dong*, *YE Zi-Ming*, *CHEN Li*, *YU Zhong-Hua*
(College of Computer Science, Sichuan University, Chengdu 610065, China)

Abstract: The existing Text2SQL methods rely heavily on the explicit mention of tables and columns in natural language queries, which causes the accuracy rate drops sharply in real-world scenarios when the same object has different names. In addition, these methods only use the database schema to capture the domain knowledge of database modeling, but the database schema, as structured metadata, has a very limited ability to express domain knowledge. This makes it difficult even for experienced programmers to fully comprehend the domain knowledge of database modeling only from the database schema, so programmers require detailed database design documents to construct SQL statements to correctly express specific queries. Therefore, we propose a Text2SQL model that uses dictionaries to expand database schema information, which parses out words or phrases in the tables and columns, queries the dictionary

收稿日期：2023-01-28
基金项目：四川省重点研发项目(2023YFG0265)
作者简介：于晓昕(1998－)，男，黑龙江绥化人，硕士研究生，主要研究领域为自然语言处理.
通讯作者：陈黎. E-mail：cl@scu.edu.cn

to obtain the semantic interpretations of these words or phrases. These semantic interpretations and the corresponding tables or columns, combined with the tables, columns and other database schema information such as primary key, foreign key are introduced to the model to learn the application field knowledge of database modeling more comprehensively. Experiments on Spider-syn and Spider dataset illustrate the effectiveness of our method, even if the table and column names used in the natural language queries are completely different from the corresponding tables and columns in the database schema, our method can get better SQL translation results, which significantly better than the latest proposed method against synonym substitution.

Keywords: Database schema; Semantic extension; Interpretation information; Text2SQL

1 引　言

现实世界中大量数据保存在关系数据库中,用户通过应用系统提供的界面访问这些数据,应用系统开发者负责将用户请求转化成数据库查询引擎所支持的 SQL 语句. 上述数据访问模式虽然应用广泛,但受限于应用系统所提供的界面,普通用户无法灵活利用数据库中的数据,任何数据访问需求的变更必须依赖于应用系统开发者设计新的 SQL 程序,而撰写正确的 SQL 语句,即使对于有经验的开发者,也不是一项轻松的工作. 为此,Text2SQL 任务被提出[1-3],旨在为数据库系统提供自然语言查询接口,这样用户和程序员无需掌握 SQL 语言,可以直接使用自然语言表达自己的数据访问需求,由查询接口自动将自然语言查询转化成 SQL 语句. 采用自然语言表达数据库查询请求,既可以减轻程序员的工作负担,也可以为普通用户提供自然灵活的数据库访问方式,因此 Text2SQL 提出后受到学术界和工业界的广泛重视.

目前研究者针对三种场景构建了标注数据集并开展相关 Text2SQL 研究:单数据库[4]、多数据库单表[5]和多数据库多表[6]. 单数据库场景假定深度学习算法只需针对特定数据库训练 Text2SQL 模型,这种场景虽然降低了任务的复杂性,但也导致模型应用受限. 多数据库场景中,数据库模式和自然语言查询共同作为 Text2SQL 模型的输入,模型需要捕捉数据库模式和自然语言查询之间的关联,并在不同的数据库模式和自然语言查询之间泛化. 不同的是,多数据库单表假定每个数据库只有一张表,不存在嵌套查询,而多数据库多表允许一个数据库含有多张表,可以多表连接或嵌套查询,因此适用面更广,是目前研究的热点.

多数据库多表 Text2SQL 目前面临的主要挑战是捕捉自然语言查询与数据库模式之间的关联,确定 SQL 查询对象,即表名、列名. 为了应对上述挑战,研究者提出了一系列编码器方法[7-14],包括基于自然语言查询和数据库模式线性化的 BERT[15]编码方法和基于图神经网络的编码方法. 但这些方法的核心是模式链接,即建立数据库模式(表名、列名)和自然语言查询之间的对齐关系,一些方法基于字面或语义相似性建立硬对齐关系,另外一些通过数据库模式与自然语言查询之间的交互注意力建立软对齐关系. 这些严重依赖模式链接的方法虽然在 Spider[6]数据集上取得了近 70% 的 Text2SQL 准确率,但 Gan 等人[16]进一步分析了上述方法在 Spider 数据集上表现良好的原因,发现 Spider[6]数据集是有偏的,被查询数据库的表名和列名在自然语言查询中明确出现,这不代表实际应用的真实情况,普通用户很难理解数据库模式,更难准确记忆表名和列名. 为了模拟真实的应用场景,Gan 等人[16]针对 Spider[6]自然语言查询中显式提及的表名和列名,人工筛选同义词替换它们,并把这种操作称为同义词替换攻击,这样替换得到的新数据集叫做 Spider-syn. 他们在 Spider-syn 上测试了上述模型中表现最好的三个[7,9,12],发现这些模型在同义词替换攻击下 Text2SQL 效果大大下降,说明现有方法过度依赖模式链接,无法适用于真实的数据库应用场景. 针对同义词替换攻击,Gan 等人[16]分别提出了对抗训练和数据库表名列名同义词扩展两种解决方法,并实验验证了这两种方法的同义词替换鲁棒性.

实质上,在数据库应用系统开发过程中,程序员在不理解数据库建模的领域知识的情况下很难准确构造实现一个查询功能的 SQL 语句,而数据库模式作为结构化的元数据,其提供的领域知识是很有限的,有经验的程序员也必须依赖数据库详细设计文档才能撰写 SQL 语句. 同样,Text2SQL 模型仅以数据库模式和自然语言查询作为输入,也很

难生成完成相应自然语言查询的 SQL 语句. 例如，图 1 给出的数据库仅包含表“公司(收入，净收入，预算，…)”，对该数据库的自然语言查询“贵公司明年预计投入多少钱来完成业务目标?”，无论是有经验的程序员还是 Text2SQL 模型，只有在理解“预算”的含义(代表为未来发展的投入)后才能确定上述查询涉及字段“预算”，而数据库模式并不包含这样的应用背景知识.

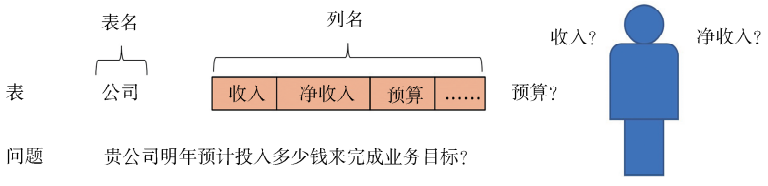

图 1　应用场景实例

Fig. 1　An example of application scenario

基于上述考虑，本文提出一种利用词典扩展数据库模式信息的 Text2SQL 方法 ExSQL(Expansion Enhanced SQL)，利用词典中包含的字段词汇语义解释，丰富数据库建模的领域知识，进一步提高 Text2SQL 模型抵抗同义词替换攻击的能力，弥补数据库模式与自然语言查询之间的语义鸿沟.

本文的主要贡献如下：(1) 提出了一种利用词典扩展数据库模式信息的 Text2SQL 方法 ExSQL，利用扩展的解释信息丰富数据库建模的领域知识，从而更准确地生成 SQL 语句. (2) 利用图神经网络来捕捉数据库模式、数据库模式词典解释文本和自然语言查询文本三者之间的语义关联，提高 Text2SQL 模型抵抗同义词替换攻击的能力. (3)在 Spider-syn 和 Spider 上进行了一系列实验，验证了 ExSQL 的有效性. 从实验数据可以发现，ExSQL 甚至优于文献[16]给出的算法的上界性能(上界性能是通过把自然语言查询中出现的表名和列名都扩展到对应的表和列上得到的 Text2SQL 结果).

2　相关工作

与人工智能的其他领域一样，目前 Text2SQL 的主流方法也是数据驱动的深度学习方法，为此研究者发布了一系列基准(benchmark)数据集，典型的有 Scholar[17]、Academic[18]、WikiSQL[5]、Spider[6]等. 它们不但为模型训练和测试提供了黄金标注数据，而且分别针对不同的应用场景定义了三种具体的 Text2SQL 任务. 数据集 Scholar[17] 和 Academic[18]针对相同数据库构造了大量自然语言查询-SQL 语句对，定义了单数据库 Text2SQL 任务，旨在训练深度学习模型捕捉固定数据库不同自然语言查询与对应的 SQL 语句之间的关系. 该任务无需在不同的数据库模式之间进行泛化，相对简单，一般的序列-序列模型往往可以取得很好的效果[19]. 但该任务要求为每个需要查询的数据库提供大量训练用的自然语言查询-SQL 语句对，并且为不同的数据库训练不同的 Text2SQL 模型，这使开发数据库自然语言查询接口成为一项费时费力的工作，适用范围受限.

数据集 WikiSQL[5] 和 Spider[6] 定义了多数据库 Text2SQL 任务. 这两个数据集均提供了多个数据库，每个数据库有多个自然语言查询-SQL 语句对，不同的是前者每个数据库仅包含单张表[5,20,21]，而后者一个数据库可能含有多张表，并且查询包含多表连接或嵌套的情况[6]. 由于多数据库多表的情况更一般，因此是目前研究的焦点.

针对多数据库多表的 Text2SQL，现有的工作普遍采用编码器-解码器架构[22]，并且在编码器中显式或隐含地堆叠模式链接器，捕捉 SQL 查询的对象-表名和列名. 在编码器中引入模式链接器的方法主要有两种，一种是基于图神经网络的方法，另一种是基于注意力的序列建模方法. 基于图神经网络的编码器[7,10-13]以数据库模式元素(表名、列名)和自然语言查询中的词(Token)作为图的结点，根据表名、列名与 Token 之间的字面相似性或语义嵌入表达相似性确定结点之间的边或作为边的权重，经过图神经网络训练后学习得到融合了自然语言查询信息的数据库模式元素表达和融合了数据库模式信息的自然语言查询表达. 基于注意力的序列编码器[14]将数据库模式和自然语言查询线性序列化(序列中包含表名、列名、主键和外键信息以及自然语言查询中的 Token)，通过 BERT 中的自注意力机制[14]捕捉序列中不同类型元素之间的软对齐关系，其中表名和列名与 Token 之间的注

意力权重隐含地实现了模式链接器的功能,或者通过字面相似性确定自然语言查询与数据库模式之间的对齐关系,然后在序列编码过程中编码这些对齐关系[14].

现有工作在设计 Text2SQL 解码器时多数采用了指针网络,区别在于一些工作将输入编码解码成 SQL 语句的抽象语法树 AST[23],另外一些直接解码成 SQL 语句的 Token 序列[14].

上述方法在 Spider 数据集上取得了大约 70% 的 SQL 语句精确匹配正确率,但是正如文献[16]指出的那样,他们强烈依赖表名和列名在自然语言查询中的显式提及,以及模式链接器对这些提及的准确捕捉. 在实际应用中,普通用户很难精确记忆数据库模式中的表名和列名. 为了验证上述方法在真实应用场景下的 Text2SQL 性能,Gan 等人[16]用同义词替换了 Spider 自然语言查询中出现的表名和列名,构建了数据集 Spider-syn,说明了现有方法在 Spider 上虚高性能的假象. 为了解决所发现的问题,Gan 等人[16]提出了两种解决方案,一种是以同义词扩展数据库模式中的表名和列名,将扩展后的数据库模式作为 Text2SQL 模型的输入,另外一种是采用对抗训练的方法,以对抗样本结合正常样本重新训练模型. 在 Spider-syn 上进行的实验发现前一种方法效果更好,并且具有不需要重新训练模型的优点.

不同于已有工作,本文认为,在数据库模式和自然语言查询之间存在语义鸿沟是自然的,这种语义鸿沟就是数据库建模的领域知识,而 SQL 查询对象(表名、列名)在自然语言查询中不出现只是这种语义鸿沟的特例,数据库详细设计文档正是程序员弥补这种语义鸿沟的重要信息来源. 但 Spider 数据集仅包含数据库模式(表名、列名、主键、外键等),每个数据库建模的领域知识缺失,更没有数据库的详细设计文档. 为此,本文提出一种利用词典扩展数据库模式信息的 Text2SQL 方法,该方法利用数据库模式中出现的 Token 的词条解释信息来弥补上述语义鸿沟,更全面地在 Text2SQL 模型中编码数据库建模的领域知识,降低现有模型对模式链接和数据库模式显式提及的依赖,提高 Text2SQL 实际应用的鲁棒性.

3　模　型

本文模型如图 2 所示,由编码器、相似度模块、图神经网络、解码器和辅助任务五部分组成.

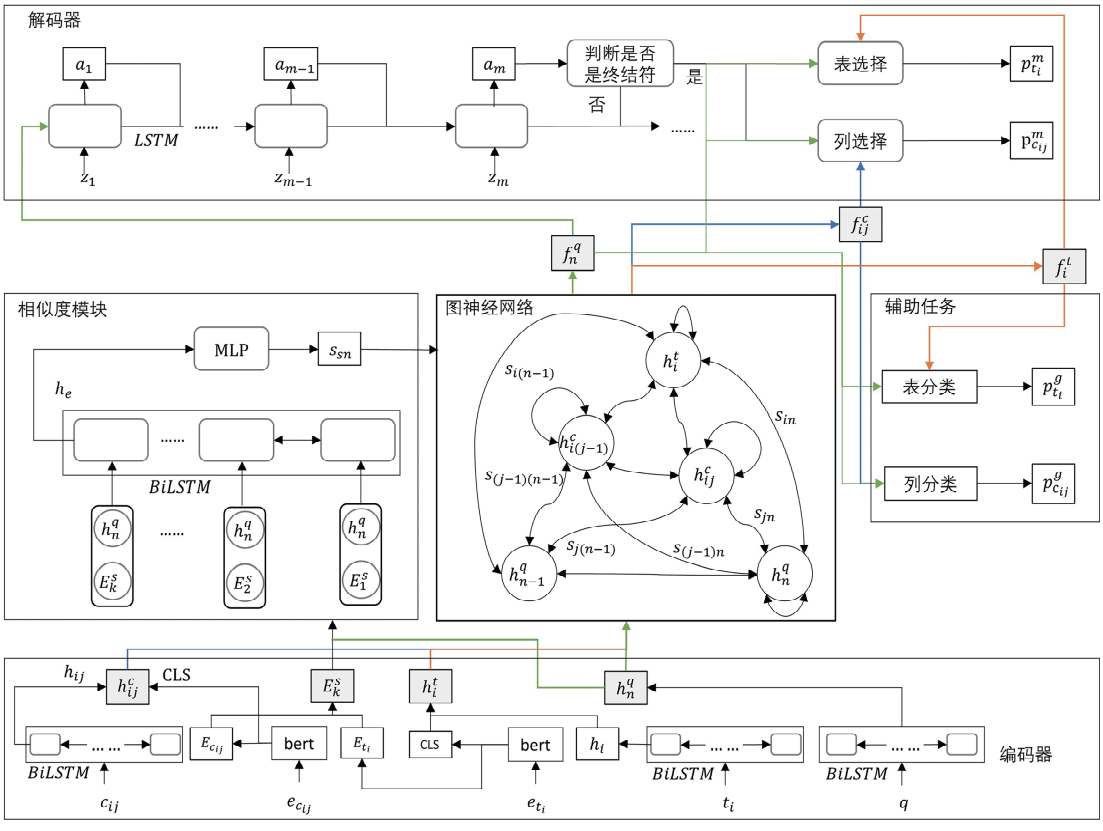

图 2　模型架构图

Fig. 2　Model architecture overview

图 2 中，编码器接收自然语言查询、数据库模式和解释信息作为输入，输出解释信息和图神经网络中结点的嵌入表达. 相似度模块利用解释信息和图中自然语言结点的嵌入表达获取图中边的权重. 图神经网络更新结点的嵌入表达后输入解码器和辅助任务中. 解码器使用有序神经元 LSTM(Ordered Neurons LSTM, On-LSTM)[24] 解码生成 AST. 辅助任务用来帮助模型学习哪些表名、列名更可能在 SQL 语句中出现.

3.1　问题定义

假定自然语言查询 $Q=\{q_1,\cdots q_n,\cdots q_{|Q|}\}$，其中 $|Q|$ 表示自然语言查询中词的个数. 表 $T=\{t_1,\cdots,t_i\cdots,t_{|T|}\}$，列 $C=\{c_{11},\cdots,c_{ij}\cdots\}$，$|T|$ 代表当前数据库中表的数量，t_i 是第 i 张表，c_{ij} 是第 i 张表的第 j 个列. 每个表名和列名可能由多个词组成，因此将 t_i 和 c_{ij} 进一步定义为 $t_i=\{t_{i1},\cdots t_{iu},\cdots\}$，$c_{ij}=\{c_{ij1},\cdots c_{ijv},\cdots\}$，其中 t_{iu} 和 c_{ijv} 是该表名和列名中的词，然后利用 t_{i0} 和 c_{ij0} 标识当前字段为一张表或描述当前列的数据类型. 解释信息 $E=\{e_k^s\}$，其中上标代表第 s 个表名或列名的解释信息，下标表示当前解释信息中的第 k 个词.

本文使用的图神经网络 $G_n=(V_n,R_n)$ 共包含三种类型的结点，$V_n=Q\cup S=Q\cup T\cup C$，其中 Q 为自然语言结点，T 为表名结点，C 为列名结点. 图中的边 $R^n=R^{S\sim S}\cup R^{Q\sim S}\cup R^{Q\sim Q}$，其中 $R^{S\sim S}$ 由预先定义的数据库模式决定，$R^{Q\sim S}$ 是通过自然语言查询和表名与列名之间的字面相似性建立起的匹配关系，$R^{Q\sim Q}$ 由自然语言查询决定.

3.2　模式信息扩展

Spider 数据集内的数据库中所有的表名和列名均使用完整英文单词命名，在词间使用下划线进行分隔. 本文将表名或列名解析为由单词组成的字符串后，枚举出所有长度为 1～3 的 gram，将其按长度降序排列成一个待查询序列，然后在 oxford 词典中查询每个 gram 的解释. 在当前 gram 的查询结果不为空时，本文会移除剩余的待查询序列中所有与当前查询对象存在重叠的 gram，当查询结果为空时，如果当前 gram 的长度大于 1 则继续向下查询，否则为了统一操作直接将当前 gram 本身作为解释使用. 查询完成后，本文将所有 gram(未被移除)的解释按顺序拼接成一段文本，作为当前数据库模式元素的解释信息，解释信息的长度通常远大于原本的表名或列名.

例如列名 book_club_id，解析出的待查询序列为：book club id；book club；club id；book；club；id；首先查询 book club id 的解释，结果为空则程序继续向下查询，book club 的查询结果不为空，所以移除 club id，book，club 三个与 book club 存在重叠部分的 gram，然后继续查询 id 的解释，最后将 book club 和 id 的解释拼接后作为当前列的解释信息.

3.3　编码器

本文将自然语言查询和数据库模式按照如下顺序组织成查询-模式序列.

$$[CLS]q_1q_2\cdots\cdots[SEP]t_{10}t_1c_{110}c_{11}c_{120}c_{12}\cdots\cdots t_{|T|0}t_{|T|}c_{|T|10}c_{|T|1}c_{|T|20}c_{|T|2}\cdots\cdots[SEP]$$

将查询-模式序列输入 BERT 后，每个词 $w=\{w_1,\cdots\cdots,w_{|W|}\}$ 被拆分为 $|w|$ 个子词，然后利用注意力层加权聚合所有子词的嵌入表达来获取该词的语义，如式(1)～式(2)所示.

$$\alpha_i=\mathrm{softmax}(\tan h(w_iv_1)v_2) \tag{1}$$

$$w=\sum_{i=1}^{|W|}\alpha_iw_i \tag{2}$$

将自然语言查询的嵌入表达，输入双向 $LSTM_1$[25] 中，拼接每个词的前向和后向的隐藏层状态，作为该词也即图神经网络中自然语言结点的初始嵌入表达 $h_n^q\in R^n$，如式(3)所示.

$$h_n^q=[\overrightarrow{LSTM}(\overrightarrow{h}_{n-1}^q,q_n);\overleftarrow{LSTM}(\overleftarrow{h}_{n+1}^q,q_n)] \tag{3}$$

图神经网络中除自然语言结点外，还包含表名和列名两种类型的结点，其初始嵌入表达均由两部分组成，二者的计算方式完全相同，在下述公式中使用 s 表示当前字段. 将 $t_i=\{t_{i1},\cdots t_{iu},\cdots\}$ 或 $c_{ij}=\{c_{ij1},\cdots c_{ijv},\cdots\}$ 的嵌入表达输入双向 $LSTM_2$ 中，将前后向的最终时刻隐藏层状态拼接后，作为表名结点或列名结点的嵌入表达的第一部分 $h_{s1}\in R^n$，如式(4)～式(5)所示.

$$h_i^s=[\overrightarrow{LSTM}(\overrightarrow{h}_{i-1}^s,s_i);\overleftarrow{LSTM}(\overleftarrow{h}_{i+1}^s,s_i)] \tag{4}$$

$$h_{s1}=[\overrightarrow{h}_N^s;\overleftarrow{h}_1^s] \tag{5}$$

对于解释信息，本文在其头尾处分别拼接 CLS 和 SEP 标识符后输入 BERT 中，得到输出的 CLS 标识符的嵌入表达作为对应结点语义表达的第二部分 $h_{s2}\in R^n$，将 h_{s1} 和 h_{s2} 拼接后利用前馈神经网络 $g(R^{2n}\to R^n)$ 将其映射回 R^n 维度，作为图中数据库模式结点的初始嵌入表达，如式(6)所示.

$$h_s=g([h_{s1};h_{s2}]) \tag{6}$$

3.4　相似度模块

在编码器输出的解释信息中所有子词的嵌入

表达E_k^s上,拼接自然语言查询中的第 n 个词的嵌入表达h_n^q后输入双向$LSTM_3$中,将前后向的最终时刻隐藏层状态拼接后输入打分函数 MLP,获得当前表名或列名与自然语言查询中第 n 个词的语义相似度得分s_{sn},计算方法如式(7)~式(9)所示.

$$E_k^{sn}=[h_n^q;E_k^s] \tag{7}$$

$$h_k^E=[\overrightarrow{LSTM}(\overrightarrow{h}_{k-1}^E,E_k);\overleftarrow{LSTM}(\overleftarrow{h}_{k+1}^E,E_k)] \tag{8}$$

$$s_{sn}=MLP([\overrightarrow{h}_N^E;\overleftarrow{h}_1^E]) \tag{9}$$

3.5　图神经网络

将编码器输出的结点的嵌入表达、对预先定义的结点间关系进行编码获得的边的嵌入表达和相似度模块输出的相似度得分三部分,作为图神经网络的初始输入. 在自然语言结点和数据库模式结点之间的边的嵌入表达上,乘以对应的语义相似性得分s_{sn},从而利用带权边来影响结点间的图注意力权重.

在本文使用的图神经网络中,结点 X 由自然语言结点、表名结点和列名结点三部分组成,定义$X=(h_n^q;h_i^t;h_{ij}^c)=\{x_1\cdots x_p\cdots x_q,\cdots,x_{|X|}\}$,其中$x_p$和$x_q$是第 p 个和第 q 个结点的嵌入表达,使用r_{pq}表示结点x_p和结点x_q之间的边的嵌入表达. 当结点x_p和结点x_q分别是自然语言结点和数据库模式结点时,s_{pq}的值在 0~1 之间,否则将对应位置的s_{pq}置为 1,通过多头注意力[26]来衡量图中结点之间的关系 α,计算方式如式(10)~式(14)所示.

$$e_{pq}^h=\frac{x_pW_Q^h(x_qW_K^h+s_{pq}*r_{pq}^K)^T}{\sqrt{d_z}} \tag{10}$$

$$\alpha_{pq}^h=softmax(e_{pq}^h) \tag{11}$$

$$\widetilde{x}_p=\mathop{\Vert}_{h=1}^{H}\sum_{q=1}^{|X|}\alpha_{pq}^h(x_q*W_V^h+r_{pq}^V) \tag{12}$$

$$\widetilde{y_p}=\mathrm{LayerNorm}(x_p+\widetilde{x_p}*W_o) \tag{13}$$

$$y_p=\mathrm{LayerNorm}(\widetilde{y_p}+FFN(\widetilde{y_p})) \tag{14}$$

在上述公式中,W_Q^h、W_K^h、$W_V^h\in R^{n*d/H}$和$W_o\in R^{n*d}$是可学习的参数,H 是注意力头数,FFN 表示前馈神经网络,LayerNorm 表示归一化层[27]. 在图迭代的过程中,每个结点x_p在聚合图中其他结点的嵌入表达时采用 H 头注意力,其中一半的注意力头只聚合来自预先定义的一跳邻居结点的嵌入表达,剩余的一半注意力头聚合来自图中所有邻居结点的嵌入表达,以此来解决图神经网络的过渡平滑问题. 经过 8 轮图迭代后,图神经网络输出所有结点的嵌入表达$f_p=(f_n^q;f_i^t;f_{ij}^c)$.

3.6　解码器

在解码的过程中利用 On-LSTM 解码生成 AST,将$z_m=[a_{m-1};a_{p_m};h_{p_m};n_m]$作为输入,得到每个时刻的隐藏层状态$h_m$,其中$z_m$中$a_{m-1}$是上一步解码生成的 action[13]、$a_{p_m}$是当前 action 的父 action、$h_{p_m}$是生成父 action 时 On-LSTM 的隐藏层状态、n_m是待拓展结点的类型,将每个时刻的隐藏层状态h_m作为查询向量聚合图中结点的嵌入表达f_p获得上下文向量 c,拼接h_m后输入 MLP 中获得h_m^{att},用于判断当前时间步如何拓展 AST,过程如式(15)~式(19)所示.

$$c_m,h_m=\mathrm{OnLSTM}(z_m,c_{m-1},h_{m-1}) \tag{15}$$

$$e_{mp}^h=\frac{h_mW_{cq}^h(f_pW_{ck}^h)^T}{\sqrt{d_z}} \tag{16}$$

$$\alpha_{mp}^h=\mathrm{softmax}(e_{mp}^h) \tag{17}$$

$$c=\mathop{\Vert}_{h=1}^{H}\sum_{p=1}^{|X|}\alpha_{mp}^hf_p \tag{18}$$

$$h_m^{att}=MLP([h_m;c]) \tag{19}$$

当待拓展的 action 不包含终结符时,使用式(20)所示的方式计算p_{a_m},否则通过式(21)和式(22)的公式计算$p_{t_i}^m$和$p_{c_{ij}}^m$,在 AST 中添加表名和列名.

$$p_{a_m}=\mathrm{softmax}(h_m^{att}W_R) \tag{20}$$

$$\zeta_{mi}^h=\mathrm{softmax}(h_m^{att}W_{tq}^h)(f_i^tW_{tk}^h)^T \tag{21}$$

$$p_{t_i}^m=\frac{1}{H}\sum_{h=1}^{H}\zeta_{mi}^h \tag{22}$$

在上述公式中,H 是注意力头数,计算$p_{c_{ij}}^m$时除参数外与计算$p_{t_i}^m$完全相同,在解码过程中使用如式(23)的负对数似然损失.

$$loss=-\sum_{p_m}\log p_m \tag{23}$$

3.7　辅助任务

辅助任务利用图中结点的嵌入表达$f_p=(f_n^q;f_i^t;f_{ij}^c)$来判断表名或列名是否会出现在 SQL 语句中,首先将f_i^t和f_{ij}^c作为查询向量聚合自然语言结点的嵌入表达f_n^q获得上下文向量c,然后将f_i^t或f_{ij}^c和c输入 biaffine[28]分类器中,计算当前表名或列名出现在 SQL 语句中的概率. 在下述式(24)~式(27)中以表名为例,除参数外列名的计算和表名完全相同.

$$e_m^h=\frac{f_i^tW_{sq}^h(f_n^qW_{sk}^h)^T}{\sqrt{d_z}} \tag{24}$$

$$\alpha_m^h=\mathrm{softmax}(e_m^h) \tag{25}$$

$$c = \left(\Big\Vert_{h=1}^{H} \sum_{n=1}^{|Q|} \alpha_{hn}^{h} f_n^q W_{sv}^{h} \right) W_{so}^{H} \tag{26}$$

$$p_{t_i}^{g} = \sigma(f_i^t W_{s_1} c^T + [f_i^t; c] W_{s_2} + b_s) \tag{27}$$

在辅助任务中，使用如式(28)的交叉熵损失.

$$loss = -\sum_{t_i} [y_{t_i} \log(p_{t_i}^{g}) + (1 - y_{t_i}) \log(1 - p_{t_i}^{g})] \tag{28}$$

4　实验与结果

4.1　实验数据

本文在两个基准数据集 Spider-syn 和 Spider 上进行模型的评估. Spider-syn 是由 Spider 衍生出的数据集，二者的数据集规模、自然语言查询所表达的语义、执行 SQL 语句的数据库完全相同. 区别之处在于，Spider 数据集中的自然语言查询和数据库模式间具有通常不会在实际的应用场景中出现的显式对应关系，而在 Spider-syn 数据集中，自然语言查询中使用的是表名和列名的同义词. 二者均包含了 7000 条训练样例，涉及到 140 个数据库，验证集中包含了 1034 条测试样例，使用在训练过程中未知的 20 个数据库. 除此之外，Spider 数据集还额外提供了从 GeoQuery[29]、Restaurants[30,31]、Scholar[17]、Academic [18]、Yelp and IMDB[32] 数据集中提取的 1965 条训练样例，共涉及 6 个数据库. Spider 数据集提供的测试集是不公开的，因此 Gan 等人[16]构建的 Spider-syn 数据集不包含测试集. 同理，本文也无法为测试集中的表名和列名扩展解释信息，在原验证集上进行模型的评估.

4.2　实验设置

本文实验中，Python 版本为 3.6.13，Torch 版本为 1.7.0. 服务器实验环境为 Ubuntu 18.04.4 LTS，搭载的显卡为 NVIIDA GeForce RTX 3090.

使用 AdamW[33] 作为优化器，将初始学习率设置为 0.1，将训练过程中的衰减率设置为 1e−5. 编码器和解码器使用的 LSTM 的丢弃率都是 0.2[34]，其余超参数见表 1.

4.3　基线模型

为了验证模型的效果，本文在 Spider-syn 和 Spider 两个基准数据集上与以下基线模型进行比较.

GNN[7]：使用图神经网络编码数据库模式，在自然语言查询和表名与列名之间通过注意力进行交互.

Global-GNN[10]：利用表名与列名的嵌入表达选择可能出现在 SQL 语句中的 top-K 后，利用自然语言查询的语义对候选表名与列名进行重排序.

表 1　模型超参数

Tab.1　Hyperparameters setting

超参数描述	超参数值
图神经网络隐藏单元个数	512
图神经网络层数	8
LSTM 隐藏单元个数	512
MLP 隐藏单元个数	512
action 嵌入维度	128
AST 中结点类型嵌入维度	128
注意力头数	8
epoch	200
batch size	20
最大梯度范数	5

IRNet[9]：在自然语言查询中划分 gram 和表名与列名进行字面匹配，辅助识别出现在 SQL 中的表名和列名，并采用基于 AST 的解码器解码.

RAT-SQL[12]：将自然语言查询和表名与列名都看成图中的结点，利用关系感知注意力辅助图编码.

BRIDGE[14]：将自然语言查询和数据库模式拼接成一个序列，利用 BERT 的自注意力机制捕捉二者之间的软对齐关系.

LGESQL[13]：利用线性图挖掘问题、表、列之间的关系，而无需事先定义元路径.

S2SQL[11]：将自然语言查询中的词间依存关系建模成图中的边，并引入解耦约束解决边的耦合问题.

RoSQL[16]：构建了 Spider-syn 数据集，并提出通过对抗训练和构建同义词词表两种方法提高模型的鲁棒性.

ETA[35]：挖掘了预训练模型学习领域基础知识的能力，结合其提出的“擦除唤醒”机制学习输入和标签之间的关联.

4.4　实验结果与分析

为了验证模型的有效性，本文和现有的基线模型进行了准确率上的对比，其中下标 L 表示 large-BERT，ManualMAS 和 AutoMAS 是 Gan 等人[16]为表名和列名扩展的同义词词表，ManualMAS 是

人工将测试集的自然语言查询中使用的同义词加入词表中,可以使模型达到实际应用场景中无法超越的性能上界,AutoMAS 是利用预训练模型自动构建的词表. 在 Spider-syn 和 Spider 数据集上的实验结果如表 2 和表 3 所示.

从表 2 实验结果可知,在受到了同义词替换攻击后,本文提出的 ExSQL 达到了最先进的性能,与 Rosql + AutoMAS 相比取得了 2.13%的性能提升(此处的 Rosql 是利用 Gan 等人[16]提出的同义词扩展方法获得的,比原论文模型性能更优异的 LGESQL). 该实验结果说明现有模型对表名和列名显式提及的依赖,而本文为表名和列名拓展的解释信息可以弥补数据库模式与自然语言查询之间的语义鸿沟,从而提高模型抵抗同义词替换攻击的能力. 本文模型性能超过了 Rosql 和 ManualMAS 同时使用达到的性能上界,说明弥补自然语言查询和数据库模式间的语义鸿沟才是 text2sql 效果的关键.

表 2　Spider-syn 数据集实验结果

Tab. 2 Experimental results on Spider-syn dataset

Model	Accuracy/%
GNN + ManualMAS	38.20
IRNet + ManualMAS	39.30
RAT-SQL$_L$	48.20
S2SQL + Grappa	51.40
Rosql + AutoMAS	58.70
Rosql + ManualMAS	59.82
ETA$_L$+CTA	60.40
ExSQL(ours)	60.83

表 3　Spider 数据集实验结果

Tab. 3　Experimental results on Spider dataset

Model	Accuracy/%
GNN	48.50
Global-GNN	52.70
IRnet	61.90
RAT-SQL	62.70
ETA	64.50
BRIDGE	65.50
LGESQL	67.60
ExSQL(ours)	67.99
S2SQL + RoBERTa	71.40

从表 3 实验结果可知,本文提出的 ExSQL 与 LGESQL 相比仅取得了 0.39%的性能提升,这是因为现有模型利用模式链接器在表名和列名被自然语言查询显式提及时,可以较为准确地判断 SQL 查询对象,因此 ExSQL 获得的提升较小. S2SQL[11]虽然利用比 BERT 效果更好的预训练模型 RoBERTa 在 Spider 数据集上达到了 71.40%的性能,但在受到了同义词替换攻击后性能大大下降. 本文的出发点即为提高模型的同义词替换攻击鲁棒性. 因此 ExSQL 在 Spider-syn 数据集上达到最先进的性能已经可以证明其效果.

4.5　消融实验

为了验证模型中各个模块的效果,本文在 Spider-syn 和 Spider 两个基准数据集上进行了一系列的消融实验,实验结果如表 4 和表 5 所示. 本文对模型中使用的各个模块进行了如下命名:

S:在自然语言查询和表名与列名之间,利用字面相似性进行模式链接.

SIM:利用自然语言查询和表名与列名的解释信息计算语义相似度得分,作为对应结点间的边的权重.

MER:在通过表名和列名获得的嵌入表达中,融合其解释信息的语义.

ExSQL-S:在 Spider 数据集上,为了更好地模拟实际的应用场景,不使用字面相似性进行模式链接.

表 4　Spider-syn 数据集消融实验

Tab. 4　Ablation experiments on Spider-syn dataset

Model	Accuracy/%
ExSQL w/o SIM	59.19
ExSQL w/o MER	59.62
ExSQL w/o AutoMAS	59.57
ExSQL	60.83

在表 4 消融实验结果中,取消 SIM 模块导致了 1.64%的性能下降,说明该模块可以在数据库模式不被自然语言查询显式提及时,帮助模型判断哪些表名和列名与当前问题更加相关. 取消 MER 模块后模型的性能下降了 1.21%,说明在数据库模式的嵌入表达中融合解释信息的语义可以弥补自然语言查询和数据库模式间的语义鸿沟,从而帮助模型在二者间建立对齐关系. 取消 AutoMAS 模块后模型的性能下降了 1.26%,说明其可以起到类似表名和列名在自然语言查询中的显式提及的

作用，也间接证明了这种显式提及确实是现有模型在 Spider-syn 和 Spider 数据集上性能差距的原因.

表 5　Spider 数据集消融实验

Tab. 5　Ablation experiments on Spider dataset

Model	Accuracy/%
ExSQL-S w/o SIM	65.09
ExSQL-S w/o MER	64.75
ExSQL-S	65.38

在实际的应用场景中，自然语言查询和数据库模式间难以建立显式的对应关系. 因此，本文在 spider 数据集上取消了利用字面相似性的模式链接器，验证本文提出的 SIM 模块和 MER 模块在这种情况下起到的作用. 在表 5 的实验中，取消 SIM 模块和 MER 模块，仅导致了 0.29% 和 0.63% 的性能下降，这说明在数据库模式被自然语言查询显式提及时捕捉 SQL 查询对象相对简单，两个模块单独使用即可得到相对较好的效果.

4.6　鲁棒性验证

为了探索在 Spider 数据集上受到破坏模式链接的攻击后，ExSQL 和其他模型的性能下降情况，本文进行了如图 3 所示的实验，图中的-S 表示取消利用字面相似性的模式链接器.

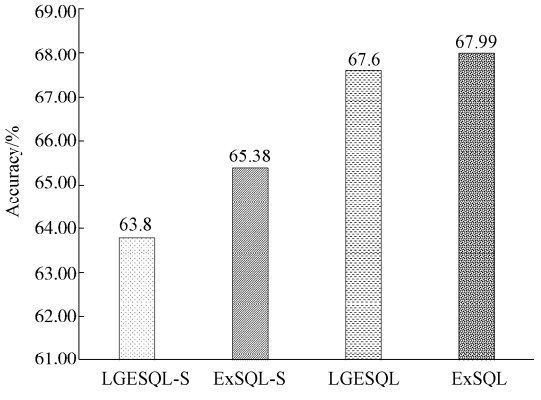

图 3　鲁棒性验证

Fig. 3　Robustness verification

ExSQL 的性能仅比 LGESQL 提高了 0.39%，然而在表 2 实验结果中，ExSQL 和达到理论性能上界的 LGESQL+ ManualMAS 相比性能提高了 1.01%. 这说明与最新提出的扩展同义词词表方法相比，本文方法具有更好的同义词替换鲁棒性. 在取消了利用字面相似性的模式链接器后，ExSQL-S 和 LGESQL-S 分别取得了 65.38% 和 63.80% 的效果，说明 LGESQL 在 Spider 数据集上取得的虚高的性能严重依赖模式链接器，而本文提出的 ExSQL 对模式链接的依赖较小，具有更好的 Text2SQL 实际应用鲁棒性.

5　结　论

针对现有方法在新的基准数据集 Spider-syn 上受到同义词替换攻击后性能下降的问题，本文提出了一种利用词典扩展数据库模式信息的 Text2SQL 方法，为表名和列名扩展解释信息来丰富数据库建模的领域知识，提高模型抵抗同义词替换攻击的能力. 通过在 Spider-syn 和 Spider 数据集上的一系列对比实验和消融实验，验证了本文模型的有效性.

本文在通用词典中为表名和列名扩展的解释信息，会不可避免地引入噪声. 例如词 mouse，在和动物领域有关的数据库中更可能表示“老鼠”的语义，而在和电脑，办公用品等领域有关的数据库中更可能表示“鼠标”的语义. 因此，下一步工作从研究的角度考虑，可以尝试利用数据库建模的领域知识对扩展的解释信息进行过滤，从应用的角度考虑，可以通过人工为数据库模式扩展更准确的解释信息，帮助模型获得更优异的性能.

参考文献：

[1] Androutsopoulos I, Ritchie G D, Thanisch P. Natural language interfaces to databases-an introduction [J]. Nat Lang Eng, 1995, 1: 29.

[2] Li F, Jagadish H V. Understanding natural language queries over relational databases[J]. Sigmod Rec, 2016, 45: 6.

[3] Affolter K, Stockinger K, Bernstein A. A comparative survey of recent natural language interfaces for databases [J]. Vldb J, 2019, 28: 793.

[4] Finegan-Dollak C, Kummerfeld J K, Zhang L, *et al*. Improving text-to-SQL evaluation methodology [C]// 56th Annual Meeting of the Association-for-Computational-Linguistics (ACL). Melbourne, AUSTRALIA: ASSOC Computational Linguistics-ACL, 2018: 351.

[5] Zhong V, Xiong C, Socher R. Seq2sql: generating structured queries from natural language using reinforcement learning [EB/OL]. [2022-12-26]. https://arxiv.org/abs/1709.00103.

[6] 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[C]//

Conference on Empirical Methods in Natural Language Processing (EMNLP). Brussels, BELGIUM: ASSOC Computational Linguistics-ACL, 2020: 3911.

[7] Bogin B, Berant J, Gardner M. Representing schema structure with graph neural networks for text-to-SQL parsing[C]// 57th Annual Meeting of the Association-for-Computational-Linguistics (ACL). Florence, ITALY: ASSOC Computational Linguistics-ACL, 2019: 4560.

[8] Lei W, Wang W, Ma Z, *et al*. Re-examining the Role of Schema Linking in Text-to-SQL[C]//Conference on Empirical Methods in Natural Language Processing (EMNLP). ELECTR NETWORK: ASSOC Computational Linguistics-ACL, 2020: 6943.

[9] Guo J, Zhan Z, Gao Y, *et al*. Towards complex text-to-SQL in cross-domain database with intermediate representation[C]// 57th Annual Meeting of the Association-for-Computational-Linguistics (ACL). Florence, ITALY: ASSOC Computational Linguistics-ACL, 2019: 4524.

[10] Bogin B, Gardner M, Berant J. Global reasoning over database structures for text-to-SQL parsing [C]// Conference on Empirical Methods in Natural Language Processing /9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Hong Kong, China: ASSOC Computational Linguistics-ACL, 2019: 3659.

[11] Hui B, Geng R, Wang L, *et al*. S2SQL: Injecting syntax to question-schema interaction graph encoder for text-to-SQL parsers[C]// 60th Annual Meeting of the Association-for-Computational-Linguistics (ACL). Dublin, IRELAND: ASSOC Computational Linguistics-ACL, 2022: 1254.

[12] Wang B, Shin R, Liu X, *et al*. RAT-SQL: relation-aware schema encoding and linking for text-to-SQL parsers[C]//58th Annual Meeting of the Association for Computational Linguistics (ACL). Virtual, Online, United states: ASSOC Computational Linguistics-ACL, 2020: 7567.

[13] Cao R, Chen L, Chen Z, *et al*. LGESQL: line graph enhanced text-to-SQL model with mixed local and non-local relations [C]//Joint Conference of 59th Annual Meeting of the Association-for-Computational-Linguistics (ACL)/11th International Joint Conference on Natural Language Processing (IJCNLP). ELECTR NETWORK: ASSOC Computational Linguistics-ACL, 2021: 2541.

[14] Lin X V, Socher R, Xiong C. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing [C]//Findings of the Association for Computational Linguistics: EMNLP 2020. Brussels, BELGIUM: ASSOC Computational Linguistics-ACL, 2020: 4870.

[15] Kenton J D M W C, Toutanova L K. BERT: pre-training of deep bidirectional transformers for language understanding[C]//Proceedings of NAACL-HLT. Minneapolis, MN, United states: ASSOC Computational Linguistics-ACL, 2019: 4171.

[16] Gan Y, Chen X, Huang Q, *et al*. Towards Robustness of text-to-SQL models against synonym substitution [C]//Joint Conference of 59th Annual Meeting of the Association-for-Computational-Linguistics (ACL). Electr Network: ASSOC Computational Linguistics-ACL, 2021: 2505.

[17] Iyer S, Konstas I, Cheung A, *et al*. Learning a neural semantic parser from user feedback [C]// 55th Annual Meeting of the Association-for-Computational-Linguistics (ACL). Vancouver, Canada: ASSOC Computational Linguistics-ACL, 2017: 963.

[18] Li F, Jagadish H V. Constructing an interactive natural language interface for relational databases [J]. Proc Vldb Endow, 2014, 8: 73.

[19] Finegan-Dollak C, Kummerfeld J K, Zhang L, *et al*. Improving text-to-SQL evaluation methodology [C]// 56th Annual Meeting of the Association-for-Computational-Linguistics (ACL). Melbourne, Australia: ASSOC Computational Linguistics-ACL, 2018: 351.

[20] Xu X, Liu C, Song D. Sqlnet: Generating structured queries from natural language without reinforcement learning [EB/OL]. [2022-12-10]. https://arxiv.org/abs/1711.04436.

[21] Yu T, Li Z, Zhang Z, *et al*. TypeSQL: knowledge-based type-aware neural text-to-SQL generation [C]// 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2018. New Orleans, United states: ASSOC Computational Linguistics-ACL, 2018: 588.

[22] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks [J]. Adv Neural Inf Process Syst, 2014, 31: 27.

[23] Yin P, Neubig G. A syntactic neural model for general-purpose code generation [C]// 55th Annual

Meeting of the Association-for-Computational-Linguistics (ACL). Vancouver, Canada: Assoc Computational Linguistics-ACL, 2017: 440.

[24] Shen Y, Tan S, Sordoni A, *et al*. Ordered neurons: integrating tree structures into recurrent neural networks [C]// International Conference on Learning Representations (ICLR). New Orleans, United States: ICLR Press, 2019.

[25] Hochreiter S, Schmidhuber J. Long short-term memory [J]. Neural Comput, 1997, 9: 1735.

[26] Vaswani A, Shazeer N, Parmar N, *et al*. Attention is all you need [C]//Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach, California: MIT Press, 2017: 6000.

[27] Ba J L, Kiros J R, Hinton G E. Layer Normalization [EB/OL]. [2022-12-20]. https://arxiv.org/abs/1607.06450.

[28] Dozat T, Manning C D. Deep biaffine attention for neural dependency parsing [C]// 5th International Conference on Learning Representations, ICLR 2017. Toulon, France: ICLR Press, 2016.

[29] Zelle J M, Mooney R J. Learning to parse database queries using inductive logic programming [C]// Proceedings of the National Conference on Artificial Intelligence. Portland, OR: Amer Assoc Artificial Intelligence, 1996: 1050.

[30] Tang L R, Mooney R J. Automated Construction of database interfaces: integrating statistical and relational learning for semantic parsing [C]// Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora held in Conjunction with the 38th Annual Meeting of the Association-for-Computational-Linguistics. Hong Kong Univ Sci & Technol. Hong Kong, Peoples R China: Assoc Computational Linguistics, 2000: 133.

[31] Popescu A M, Etzioni O, Kautz H. Towards a theory of natural language interfaces to databases[C]// Proceedings of the 8th International Conference on Intelligent User Interfaces. Miami Beach: Association for Computing Machinery (ACM), 2003: 327.

[32] Yaghmazadeh N, Wang Y, Dillig I, *et al*. SQLizer: query synthesis from natural language[J]. P ACM Program Lang, 2017, 1: 1.

[33] Loshchilov I, Hutter F. Decoupled weight decay regularization [C]//International Conference on Learning Representations. New Orleans, United States: ICLR Press, 2018.

[34] Gal Y, Ghahramani Z. A theoretically grounded application of dropout in recurrent neural networks [C]//Proceedings of the 30th International Conference on Neural Information Processing Systems. Barcelona, SPAIN: MIT Press, 2016: 1027.

[35] Liu Q, Yang D, Zhang J, *et al*. Awakening Latent grounding from pretrained language models for semantic parsing [C]//Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. Virtual, Online: ACL, 2021: 1174.