清 华 大 学

综 合 论 文 训 练

题目：自然语言到多类型查询语言的翻译方法研究

系　　别：软件学院

专　　业：软件工程

姓　　名：任家纬

指导教师：王朝坤　副教授

2024 年 6 月 10 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：伍家炜　导师签名：王朝坤　日　期：2019年5月27日

中文摘要

在大数据时代，多类型存储系统受到日益增多的关注和愈加广泛的使用。复杂的多类型查询语言使得无论对于普通用户还是专业开发人员，查询多类型存储系统都并非易事。为便于多类型存储系统的用户快速构建查询语句，文章提出了一种自然语言处理任务“自然语言到多类型查询语言的翻译”，旨在将自然语言转换为多类型查询语言，以支持多类型存储系统。文章构建了适用于“自然语言到多类型查询语言的翻译”任务的基准数据集和基线方法。同时，提出了一个面向多类型查询语句的类型和结构信息的方法，设计了基于 Encoder-Decoder 架构的多类型语言翻译模型，在 Encoder 部分注入查询语句涉及的模型类型信息，并在 Decoder 部分利用查询语句结构特征，以提升模型效果。在多个数据集上的实验结果表明，文章提出的方法显著优于基线方法，从而证实了其有效性。文章还提出了 NL2PQL 任务未来发展的多个方向，包括多模态支持、跨语言支持、实时交互和反馈机制等。这项工作对多类型存储系统的查询生成具有重要意义。

关键词：自然语言到多类型查询语言的翻译；多类型存储系统；机器翻译；自然语言处理；深度学习

ABSTRACT

In the era of big data, polystore systems are receiving increasing attention and wider adoption. Complex polystore query languages pose challenges for both ordinary users and professional developers in querying polystore systems. To facilitate users of polystore systems in quickly constructing query statements, this paper proposes a natural language processing task, natural language to polystore query language(NL2PQL), aiming to convert natural language into polystore query language to support polystore systems. The paper constructs a benchmark dataset and baseline methods for the natural language to polystore query language task. Additionally, it introduces an innovative approach focused on the scope and skeleton information of polystore query statements. This involves designing a neural network model based on an Encoder-Decoder architecture, injecting model scope information related to the query statements into the Encoder part, and utilizing skeleton features of the query statements in the Decoder part to enhance model performance. Experimental results on multiple datasets demonstrate that the new method significantly outperforms baseline methods, thereby confirming its effectiveness. The paper also suggests several future directions for the NL2PQL task, including multi-modal support, cross-language support, real-time interaction, and feedback mechanisms. This work is of significant importance for query generation in polystore systems.

Keywords: NL2PQL; polystore; machine translation; natural language processing; deep learning

目　录

第 1 章　引言

1.1　选题背景及意义

在大数据时代，数据库领域的数据模型和数据管理系统经历了快速增长，每个系统都针对特定垂直市场[1]进行了优化。这些系统充分体现了“no one size fits all”的格言[2]。

对于典型的结构化数据，将其存储在关系型数据库中是合理的选择。然而，对于大量的半结构化和非结构化数据，如文档数据、图数据等，继续采用关系型模型会导致大量的软硬件开销和资源浪费。为了解决这一问题，以适合的方式在不同引擎上存储具有不同性质的数据，多类型存储（polystore）系统应运而生[3]。

如今，数据库管理系统（database management system, DBMS）的应用已从传统的银行等公司扩展到几乎所有行业。DBMS 的用户范围也从经过良好培训的数据库专家扩展到可能不熟悉底层数据库模式、甚至不了解基本 DBMS 概念的数据分析师和普通用户。Polystore 系统作为一种 DBMS，与常规 DBMS 一样对用户是透明的，用户可以通过数据库查询语言与系统进行交互。与结构化查询语言（structured query language，SQL）对应，本文将 polystore 系统的查询语言称为 PQL（polystore query language，PQL）。目前已有的 polystore 系统如 BigDAWG[3]、ArangoDB、OrientDB、Polypheny[4]等，都各自设计了自己的 PQL。图1.1是 BigDAWG[3]的一个 PQL 查询示例：

可以发现，PQL 的结构复杂，语法多样，通常结合了多种数据模型对应的语言的特征。初学者可能需要花费大量时间和精力去掌握各种数据存储模型以及相应的查询语言。即便是专业开发人员，编写一条合理的查询语句也是一项挑战。为了方便 polystore 系统的用户快速构建查询语句，执行后续 polystore 系统的使用、检查或维护工作，一个直接的想法是寻找一种将自然语言翻译为 PQL（natural language to polystore query language，NL2PQL）的方法，以便用户可以使用自然语言描述他们的查询意图，并依赖 NL2PQL 方法成功将其转换为正确的 PQL 查询语句。

```
bdrel(
  select *
  from bdcast(
    bdtext({
      'op': 'scan',
      'table': 'logs'
    }),
    tab1,
    '(cq1 text, mimic_text text)',
    relational
  )
)
```

图 1.1　BigDAWG 的查询语句示例

1.2　相关工作

构建数据库查询语句的目的通常多样而复杂，包括但不限于：（1）检索数据库以获取所需数据；（2）通过边界情况检查查询处理中的逻辑问题；（3）比对语义等价的查询语句，以分析数据库系统的优化效果。在明确目的且领域单一的任务中，通常使用基于文法的方式生成查询语句。在复杂情景下，通常采用深度学习的机器翻译方法。然而，当前面向不同任务的方法大多聚焦于关系型数据库，使用 SQL 作为查询语言，仅有少数任务关注图数据库 Neo4j 和文档数据库 MongoDB，针对 polystore 领域的方法尚未得到充分研究，仍存在广阔的研究空间。

1.2.1　自然语言到结构化查询语言的翻译

自然语言到查询语言的翻译工作，最早围绕自然语言到结构化查询语言的翻译（natural language to structured query language，NL2SQL）展开。早期数据库社区的研究人员利用基于规则的方法[5-7]解决这一问题，先将自然语言问题解析为中间表示（如解析树），再利用规则将其映射为 SQL 语法树，最终转成 SQL 查询。接着出现了关于数据集的研究, 例如 ATIS[8]、GeoQuery[9]、Scholar[10]、Academic[11]、Advising[12]、Restaurants[13]等. 但由于这些数据集仅关注单一领

域, 而且样本数太少, 所提出的研究方法即使在特定领域表现良好, 但通常不具备泛化能力和通用性[14]。近些年，来自深度学习（deep learning，DL）和自然语言处理（natural language processing，NLP）社区的深度学习方法，提供了解决跨领域问题的新的方式。跨领域的大型数据集 WikiSQL[15]和 Spider[16]相继提出，在为 NL2SQL 制定更复杂更困难的任务的同时，也推动着研究人员们不断尝试新的方法。大多数现有工作将 NL2SQL 任务视为一个语义解析任务并通过序列到序列（sequence to sequence，Seq2seq）的模型解决[17-19]。编码器（Encoder）部分，往往是 LSTM[20]或者 Transformer[21]架构；解码器（Decoder）部分，使用基于草图的槽位填充方法较为有效[22-23]。随着大规模预训练模型（如 BERT[24]、T5[25]等）的提出，研究人员们将预训练引入 NL2SQL 问题，在各个数据集上效果都有了显著的提升。此外，图神经网络（graph neural network，GNN）在 NL2SQL 任务中也有着不俗的表现[26]。

1.2.2 自然语言到图数据库查询语言的翻译

Cypher 是图数据库 Neo4j 的查询语言。Cypher 复杂的操作和语法通常需要用户付出比 SQL 更高的学习成本。因此，与 SQL 数据库的需求类似，需要一种语义解析技术（类似 NL2SQL），来帮助用户将自然语言查询自动转换为 Cypher。Cypher 和 SQL 一样，是一种声明式文本查询语言，包含语句、关键词和表达式等要素，如谓词、函数等（WHERE、ORDER BY、SKIP LIMIT，AND，p.unitPrice > 10）。但与 SQL 不同，Cypher 表示图模型，因此添加了一个特殊的子句 MATCH 来匹配数据中的这些模式，并使用括号表示节点实体（()），箭头表示有向连接（–>），方括号表示属性（()–[:ORDERED]->()）。为此，研究人员们制定新的自然语言到 Cypher 的翻译任务，并为其标注了数据集 SPCQL[27]。然而，大多数现有的 NL2SQL 领域的最先进的模型无法简单迁移到自然语言到 Cypher 的领域，后续研究仍有待展开。

1.2.3 面向数量约束的查询语句生成

生成查询语句的有助于提高用户对数据库的使用效率，并且方便维护者检查数据库系统中的错误，从而进行优化。LearnedSQLGen 框架利用强化学习来解决带有约束的 SQL 查询生成问题。该框架通过设计奖励函数来引导查询生成过程，以满足特定的约束（如结果大小或执行代价），并采用有限状态机来确保生成的 SQL 查询在语法和语义上正确[28]。

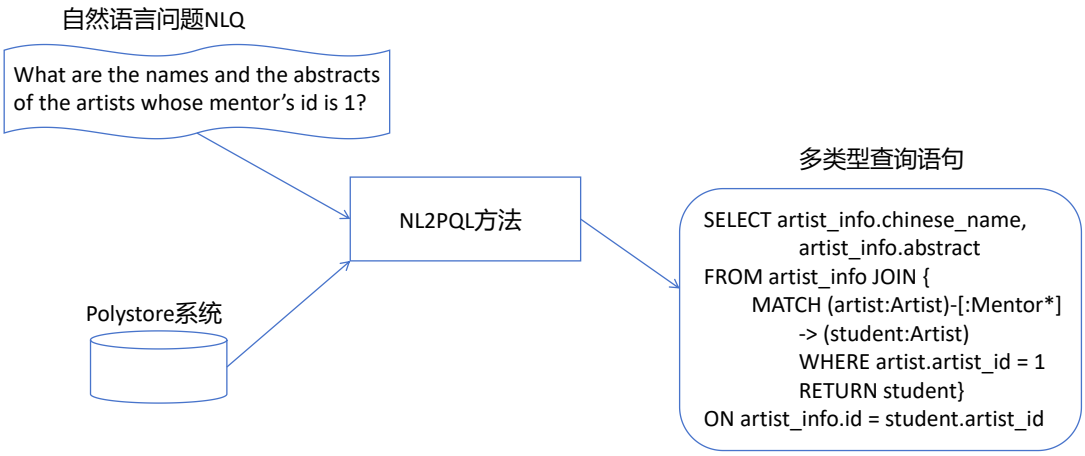

图 1.2　自然语言到多类型查询语言任务示例

1.2.4　面向逻辑检查的查询语句生成

近年来，许多来自数据库社区面向 DBMS 逻辑错误检查的工作设计了不同的查询语句生成方法。QPG 方法利用查询计划指导查询语句生成，根据查询计划反映数据库系统内部执行逻辑的假设，并通过探索更多独特的查询计划来提高发现逻辑错误的可能性，通过逐步变异数据库状态来生成更多独特的查询计划[29]。TQS 方法通过构建模式图，在模式图上随机游走选择表，再随机选取查询条件生成完整的查询语句[30]。这些方法的共同特征是只能按设定结构生成 SQL 语句，难以推广到其他任务。

1.3　任务定义

由于 NL2PQL 是一个全新的任务，本文首先将对其进行明确的定义。简单来说，用户提供一个自然语言问题（natural language question，NLQ），以及对应 polystore 系统的模式信息（schema），通过 NL2PQL 方法，能够生成对应自然语言问题的 PQL。

数据库模式 以字母 $\mathcal{D}$ 标识 polystore 系统中的数据库，数据库中的数据可能以多种模型存储，如关系、图、文档等，这里用 scope 标识数据的存储模型。数据库模式 S 包含（1）N_1 张以关系型存储的表 $\mathcal{T}_1 = \{t_1, t_2, ..., t_{N_1}\}$，$N_2$ 种以图模型存储的节点或关系（表、节点、关系以及各种不同数据模型的数据库中的存储数据结构统称为**实体**）$\mathcal{T}_2 = \{t_1, t_2, ..., t_{N_2}\}$，以及 $N_n (n \geqslant 2)$ 种其他任意模型存储的实体，总共 $N = \Sigma_i^n N_i$ 个实体；（2）对于每个实体 $t_i (1 \leqslant i \leqslant N)$，包含一组

列或者属性（统称为**属性**）$C_i = \{c_1^i, ..., c_{n^i}^i\}$，其中 n^i 是第 i 个实体包含的属性总数。（3）可跨模型连接的一组外键 $\mathcal{R} = \{(c_a^i, c_b^j) | c_a^i \in C_i, c_b^j \in C_j, i \neq j\}$。

自然语言问题 自然语言问题是以英文描述的，语义清晰，不含歧义的问题。歧义指的是含义不清晰、无法量化的情况。例如，问题“Who is the best student in the class?”是不恰当的，因为“the best”无法定义。一个合适的自然语言问题应该像是“Who has the highest math score in the class?”或是“Who is the tallest in the class?”。

NL2PQL 任务 形式上，给定一个自然语言问题 q 和一个数据库 D 以及对应模式信息 S，NL2PQL 任务的目标是将 q 翻译成能在 D 上执行的 PQL ℓ。

1.4 研究方法

本文将 NL2PQL 任务视为一种机器翻译任务，尝试使用深度学习的方法来解决问题。借鉴 NL2SQL 任务的研究经验，本文将 NL2PQL 任务分解为三个主要问题：首先，构建包含“自然语言问题-多类型查询语句”序列对的数据集，以作为基准测试；其次，提供适用于在数据集上进行训练和测试的基线方法，以进行对比分析；最后，研究和设计比基线方法更优的新的深度学习方法。

多年来，研究人员为 NL2SQL 任务标注了大量规模和难度不一的数据集，如 WikiSQL[15]，Spider[16]、Advising[12]等。收集数据库和构建相应的自然语言问题是一项耗时且繁琐的工作。鉴于人力和时间资源的限制，本研究将通过改造现有 NL2SQL 数据集的方法来构建 NL2PQL 任务所需的基准数据集。同时，本研究还将基于一些现有的 NL2SQL 方法，将其转化为适用于 NL2PQL 任务的基线方法。

1.5 本文贡献

本文的贡献如下：

- 提出了自然语言到多类型查询语言（NL2PQL）的任务；
- 适配了 NL2PQL 任务的基准数据集和基线方法；
- 提出了解决 NL2PQL 任务的面向多类型查询语句的类型和结构信息的翻译方法。

1.6 论文结构

本文的余下部分将按照以下结构组织：第二章介绍数据集和基线方法的适配。第三章介绍本文提出的面向多类型查询语句的类型和结构信息的翻译方法。第四章介绍实验的设置和结果。第五章介绍本文搭配实验设计的的可视化系统。第六章是全文的总结部分。

第 2 章　NL2PQL 任务基准数据集和基线方法的适配

本章首先介绍本研究使用的多类型查询语言，随后详细描述构建 NL2PQL 任务数据集的过程以及改进后的结果，最后介绍从 NL2SQL 任务适配的三个基线方法。

为了标注 NL2PQL 任务的数据集，本文从已发展多年的 NL2SQL 任务出发，首先从网络上获取了 NL2SQL 任务的已有数据库和标注好的数据集[12,15-16]，并选取了其中的 WikiSQL[15]、Advising[12]、Restaurants[13]和 IMDB[31]四个能直接获取数据库的数据集，在已有的 NL-SQL 标注的基础上，进行 NL-PQL 标注的转换。

本文将 NL2PQL 任务视作序列到序列的机器翻译问题，选取了 Dong 等人提出的 Seq2seq 和 Seq2seq+attention[32]作为基线方法并进行了部分的改动，将输出空间限制为数据库模式、自然语言问题与 PQL 关键词的并集。此外，Zhong 等人提出的 Seq2SQL 模型[15]在 WikiSQL 数据集上有着十分可观的表现，本文对其进行了必要且有限的改动，使其能够在更改过的 NL2PQL 数据集上运行，将其作为另一个基线方法。

2.1　多类型查询语言

由于将 NL2SQL 任务的数据集和基线方法适配到 NL2PQL 任务上是一项极为复杂和困难的任务，通常需要耗费大量的人力和时间成本。因此，本文采用了一种简化但可扩展的方法进行研究。本文使用的 PQL 仅考虑了关系型和图两种存储模型，涉及了 SQL 和 Cypher 两种查询语言。具体来说，如果查询涉及的实体（表、节点、关系）存储在单一模型中，本文可以根据实体的 scope 属性决定使用 SQL 或 Cypher 来编写查询语句。若涉及跨模型的查询，在连接实体时，本文根据 scope 信息决定某一部分使用 SQL 还是 Cypher，并用大括号（{}）分隔不同实体的查询语句。从图 2.1 给出的示例可以观察到，跨模型的查询语句仍然保留了原查询语句的特征，只是在涉及实体连接的部分使用大括号进行了跨模型连接。这种 PQL 的设计可以轻松扩展到更多类型的存储模型和查询语言，只需增加大括号（{}）的嵌套即可。

```
SELECT artist_info.chinese_name, artist_info.abstract
FROM artist_info
JOIN {
        MATCH (artist:Artist)-[:Mentor*]-> (student:Artist)
        WHERE artist.artist_id = 1
        RETURN student
        }
ON artist_info.id = student.artist_id
```

(a) SQL 内嵌入 Cypher

```
MATCH (follower:User)-[:Follow]->(long_time_user:User),
{ SELECT * FROM user } AS sql
WHERE long_time_user.user_id = sql.user_id
AND sql.register_time <= '2018-12-10 08:00:00'
RETURN follower.user_id
```

(b) Cypher 内嵌入 SQL

图 2.1　PQL 示例

2.2　数据集

本文研究了一系列 NL2SQL 任务常用的数据集，包括 Spider[16]、WikiSQL[15]、Advising[12]、Geography[8]、ATIS[8]等，并对各个数据集的规模、查询语言难度以及数据库模式复杂程度进行了分析。最终，从中挑选了难度和规模各不相同的四个数据集作为基础，并将它们转换为风格统一的 NL2PQL 任务的基准数据集。

WikiSQL　WikiSQL[15]是目前 NL2SQL 任务中规模最大的数据集，包含 24241 张表和 80654 个自然语言问题，每张表都单独存储在一个数据库当中，每个查询语句均为简单的“select ... from ... where ... ”结构，其中包含聚合函数和多个查询条件。WikiSQL[15]的数据来自于维基百科，因此其中的自然语言问题都符合现实逻辑。

Advising　Advising[12]的问题数据集收集自密歇根大学的课程信息数据库，但存在虚构的学生记录。其中一些问题是从电气工程与计算机科学的院系主页收集的，另一些问题是由了解数据库的计算机系学生编写的。自然语言问题和对应的查询语句都经过多人的校验与核对。

表 2.1　适配后的 WikiSQL、Advising、Restaurants、IMDB 数据集情况

数据集	问题-查询对数量	关系实体数量	图实体数量	数据库数量
WikiSQL	80654	53770	26884	24241
Advising	7449	9	1	1
Restaurants	125	2	1	1
IMDB	142	13	3	1

Restaurants　Restaurants[13]的数据包含一些美国的餐馆的信息，包括它们供应的食物品类以及所处位置。

IMDB　IMDB[31]收集来自 Yelp 网站和网络电影数据库的信息，数据库规模庞大，NL2SQL 数据集中只使用了其中一小部分数据。

原始的 NL2SQL 数据集中的查询语句仅涉及 SQL。因此，在构建数据集时，本文选择了一部分存储在关系数据库中的表，将其转而存储在图数据库 Neo4j 上。根据这些表之间的外键关系以及 SQL 中 JOIN 语句的连接方式，本文决定将部分表转换为 Neo4j 中的节点（node），部分表转换为关系（relationship）。与关系型数据相比，图数据之间的关系更加灵活多样，节点和关系之间的连接也具有逻辑内涵。因此，将关系数据转换为图数据的选择并非随意的过程，应当选择那些在关系型数据库中存在外键或类似外键关系的数据进行转换。例如，在 IMDB 数据库中有 16 张表，本文选择了三张存在逻辑关联的表 Movie、Director、Directed_By 存储到图数据库中，将 Movie 和 Director 存储为节点，将 Directed_By 存储为关系。

具体来说，本文使用正则匹配的方法找到了每个 SQL 中被存储到 Neo4j 的实体以及与其相关的查询条件，用 Cypher 改写后进行替换。用程序完成批量改写后，根据每个数据集本文编写了运行 PQL 的脚本，测试本文改写后的 PQL 是否都能成功运行。值得一提的是，原数据集中就存在着少量 SQL 语句因为语法错误无法执行，对于这部分语句本文将其移出数据集。此外，本文通过人工审查的方式重点查看了涉及嵌套的查询，特别是复杂的嵌套，将一些程序改写错误的 PQL 手动纠错，完成数据集的构造。适配到 NL2PQL 任务后的数据集，每条数据包含一个自然语言问题以及对应的多类型查询语句。查询语句有三种形式，包括纯 SQL 的查询语句、纯 Cypher 的查询语句以及 SQL 和 Cypher 互相嵌套的查询语句。

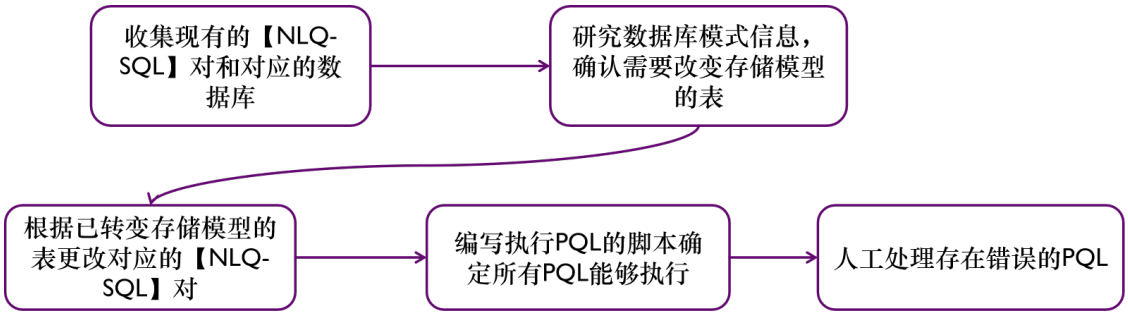

图 2.2　NL2PQL 数据集准备过程

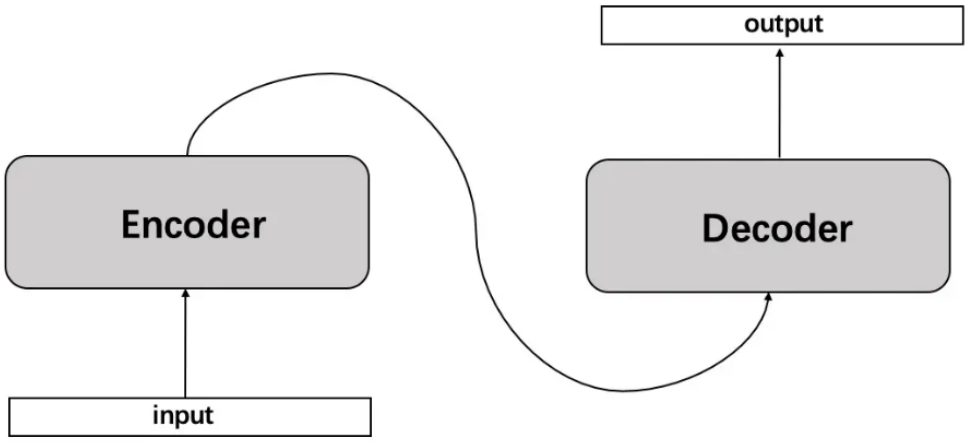

图 2.3　Encoder-Decoder 架构

需要注意的是，本文只处理了无法执行的 PQL 语句，数据集中仍可能存在自然语言问题语义模糊、自然语言问题与查询语句不匹配、自然语言问题重复等问题。

表2.2展示了四个数据集中的查询语句。观察到，这些查询语句的复杂程度各不相同。WikiSQL[15]数据集不涉及多表查询，因此仅包含纯 SQL 或纯 Cypher 语句。Advising[12]数据集和 Restaurants[13]数据集中存在 SQL 与 Cypher 嵌套的查询语句，但 Cypher 部分仅以“节点”的形式出现，没有涉及“关系”的形式。IMDB[31]数据集的查询语句最为复杂，不仅涉及多模型连接，同时 Cypher 部分还包含“节点-关系”的形式。

值得注意的是，查询语句的复杂程度并不是衡量数据集的难度的唯一标准。评估一个数据集的难度，还需要考虑数据集的规模、数据库的数量、数据集中查询语句的相似程度以及自然语言问题的语义清晰程度。因此，综合表2.1和表2.2的内容，四个数据集难度从大到小依次是 WikiSQL[15]、IMDB[31]、Restaurants[13]、Advising[12]。

表 2.2　各个数据集的查询语句示例

数据集	多类型查询语句 PQL
WikiSQL	match (n:table) where n.player = terrence ross return n.nationality select count(school/club team) from table where player = jalen rose
Advising	select distinct course.department , course.name , course.number , program_course.workload from course join { match (area:area) return area } on course.course_id = area.course_id join program_course on program_course.course_id = course.course_id where area.area like ”%theory%” and program_course.workload < (select min(program_course.workload) from course join program_course on program_course.course_id = course.course_id where course.department = ”ENGR” and course.number = 490) order by program_course.workload
Restaurants	select location.house_number , restaurant.name from { match (geographic:geographic) where geographic.region = ”yosemite and mono lake area” return geographic } join location on geographic.city_name = location.city_name join restaurant on location.restaurant_id = restaurant.restaurant_id where restaurant.city_name = geographic.city_name and restaurant.food_type = ”french” and restaurant.rating > 2.5
IMDB	select count(distinct (actor.name)) from actor , cast , { match (movie : movie) - [directed_by : directed_by] -> (director : director) where director.name = ”jim jarmusch” return movie.mid } where actor.nationality = ”iran” and cast.aid = actor.aid and movie.mid = cast.msid

2.3　基线方法

2.3.1　Seq2seq

序列到序列（Sequence to sequence，Seq2seq）是用于自然语言处理的一族机器学习方法，对应的神经网络模型通常具有如图2.3的 Encoder-Decoder 架构。

2.3.1.1　朴素的 Seq2seq

Encoder-Decoder 是一种通用框架，其中 Encoder 和 Decoder 部分均可以处理文字、图像、语音、视频等序列化数据。在用于机器翻译任务的 seq2seq 模型中（图2.4），编码器（Encoder）接收一个序列，将其编码成一个固定长度的向量，通常采用循环神经网络（RNN）或其变体如长短期记忆（LSTM[20]) 实现。在编码过程中，编码器将输入序列的每个元素依次输入到神经网络中，最终得到一个包含整个序列信息的向量。解码器（Decoder）将编码器输出的向量解码成一个序

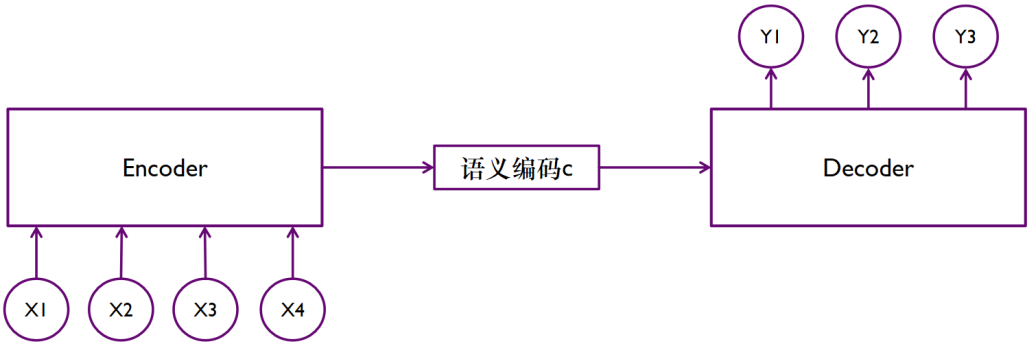

图 2.4　Seq2seq 模型架构

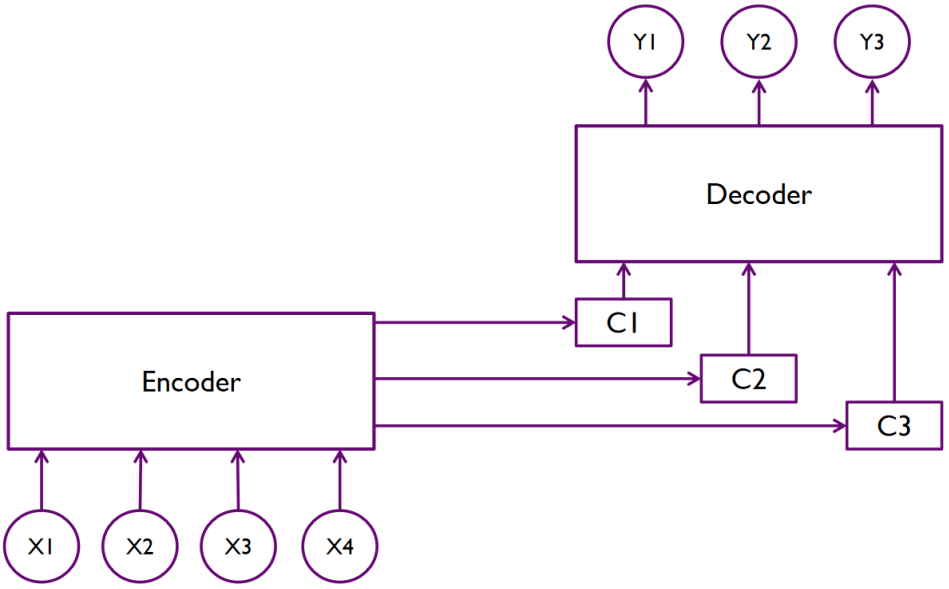

图 2.5　Seq2seq + attention 模型架构

列，同样采用 RNN 或 LSTM 实现。在解码过程中，解码器首先接收编码器输出的向量，然后逐步生成目标序列的元素。在每个时间步，解码器根据上一个时间步的输出和编码器输出的向量，计算当前时间步的输出。

2.3.1.2　Seq2seq+attention

朴素的 Seq2seq 模型的编码器实际上将所有信息加权平均传送给了解码器，但更为合理的做法应当是解码器在每个时间步上重点关注这个时间步附近编码器的信息。引入了注意力（attention）机制的 Seq2seq 模型（图2.5）解决了这一

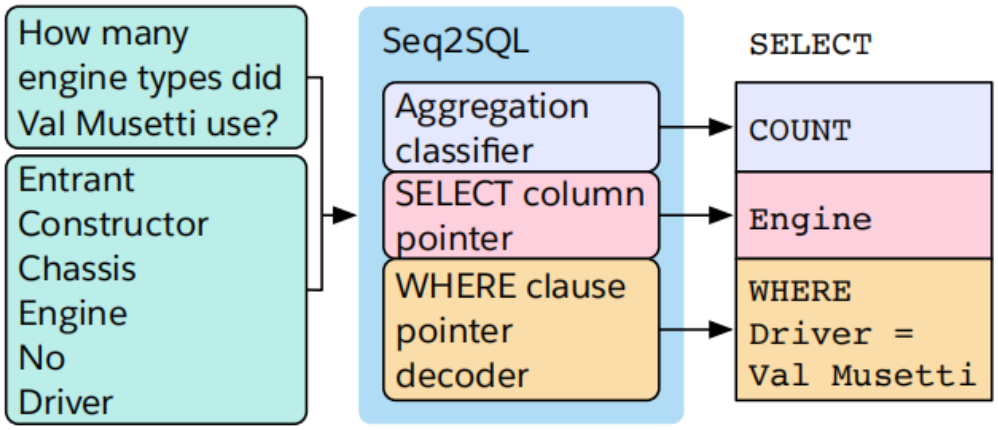

图 2.6　Seq2SQL 模型架构

点。使用注意力后，解码器的每个时间步都能接收编码器信息，并且能接收与该时间步关系最密切的信息，从而生成更合理的输出。

适配方法　Dong 等人使用 LSTM[20]实现 Encoder 和 Decoder，而本文将 Encoder 和 Decoder 均实现为双向 GRU[33]，添加了注意力机制的版本则在 Decoder 部分增加注意力机制。词表被设置为数据库模式、自然语言问题与 PQL 关键词的并集。词嵌入并未使用通过 Word2Vec[34]或者 GloVe[35]方法预训练得到的嵌入，而是简单地随机初始化。

2.3.2　Seq2SQL

Seq2SQL 模型（图2.6）和数据集 WikiSQL 在同一研究工作中提出[15]，借鉴了 Seq2seq 模型中的 Encoder-Decoder 架构，并在 SQL 的查询条件部分引入了强化学习模块进行解码，在 WikiSQL 数据集上达到了 59.4% 的执行正确率。Seq2SQL 模型主要由三个部分组成：聚合函数分类（aggregation classifier）、列名选取（select column）和条件解析（where clause）。聚合函数分类主要负责确定 SQL 查询中使用的聚合函数（如 min、max 等），列名选取部分本质上也是一个分类器，负责选择概率最高的列。条件解析部分采用了强化学习的方法，因为直接使用字符串的字面量计算损失函数会将条件顺序不同的情况视为错误，从而加入损失；使用强化学习的策略梯度（policy gradient）能够有效避免这一问题。

适配方法 在适配 Seq2SQL 模型的过程中，本文没有修改模型的总体架构，而是在每个部分预设的字符集和涉及 SQL 关键字判断的代码进行了调整，使之能够识别 Cypher 关键字，从而将原模型对 SQL 的识别扩展到本文使用的 PQL。同时本文将执行 SQL 的代码修改成了执行 PQL 的代码，确保确保模型能够在本文提出的数据上进行训练和推理。

第 3 章　面向多类型查询语句的类型和结构信息的翻译方法

近些年来，研究人员在解决 NL2SQL 任务上常使用预训练语言模型（pre-trained language model，PLM）。PLM 是一种通过大规模文本数据进行训练的人工智能模型，在预训练阶段，模型通过阅读大量文本数据，学习语言的统计规律和语义信息。这种预训练过程使得模型能够理解语言的结构、语法、语义以及上下文关系。PLM 通常采用无监督或半监督的方式进行训练，即在没有具体标签或任务指导的情况下学习。训练完成后，通过微调（fine-tune）或迁移学习（transfer learning），往往可以在不同的自然语言处理任务上取得较好的结果。

现有大量的预训练模型可用于解决自然语言处理问题。BERT[24]使用双向 Transformer[21]的 Encoder 结构，通过在预训练阶段同时考虑文本序列中的前后文信息来生成上下文相关的语言表示，在自然语言理解任务上表现出色。GPT[36-37]基于 Transformer[21]构建，采用单向语境处理，在预训练阶段利用文本序列中当前位置之前的信息来预测下一个词，在自然语言生成任务上表现优异。Grappa[38]是专为 NL2SQL 任务设计的预训练模型，通过同步上下文无关语法 (SCFG) 在高质量的表上构造合成的“自然语言问题-SQL”对用于预训练，并使用一种新的文本-模式链接（text-schema linking）目标来预测每个对的 SQL 中表列的语法角色，以此鼓励模型识别可以在 SQL 中用到的模式成分。T5[25]基于 Transformer[21]构建，以一种统一的方式处理各种 NLP 任务，将所有任务都转化为文本到文本的问题，T5 的核心思想是将输入和输出都表示为文本序列，这使得模型在各种任务上具有更大的通用性和灵活性。

上述几种模型以及其他未提及的预训练模型在 NL2SQL 任务中均有大量应用。针对 NL2PQL 任务，本文提出的面向多类型查询语句的类型和结构信息的翻译方法选择使用通用性更强的 T5[25]作为基础的预训练模型，通过对模型的微调完成翻译任务。

3.1　模型设计思路

在使用通用机器翻译模型和 NL2SQL 模型来解决 NL2PQL 任务时，模型的表现效果并不理想。一个典型的错误是模型无法正确预测涉及某个实体的位置应该使用 Cypher 风格的语法还是 SQL 风格的语法。例如，对于一个 PQL 查询语

句：“match (n:1_10015132_16) where n.name = ”ohau school” return min(n.decile)”，其中实体“1_10015132_16”以关系型模型存储，但模型错误地预测了查询语句应该以 Cypher 风格来编写。

在处理更复杂的 PQL 时，另一个问题是生成的查询语句无法正确判断 SQL 和 Cypher 嵌套的位置，导致 PQL 语法错误。例如，“select distinct ... from ... where match (a:area) return a as areaalias0 (areaalias0.area ...)”。模型错误地判断了 match 子句的位置，导致语法上的错误。

此外，由于 PQL 的关键字集合相对复杂，模型在推理阶段可能会随机编造字符，这放大了模型在生成 PQL 时引入无关属性和实体的可能性。严重情况下，这种问题会导致 PQL 的执行时间延长至数小时之久，难以在短时间内验证生成的 PQL 的执行正确性。

为了解决上述提及的问题，本文设计了面向多类型查询语句的类型和结构信息的多类型语言翻译模型，Scope-aware Encoding & Skeleton-guided Decoding（SESD）（见图3.1）。具体来说，类型信息指的是某个查询语句所涉及的实体的 scope 信息，结构信息指的是由 PQL 关键字组成的结构特征，如“select _ from _ match _ return _ where _”，即 skeleton 信息。

该模型基于 Encoder-Decoder 架构，在 Encoder 部分，向模型注入数据库的 scope 信息，以排除无关的属性和实体，减轻模型在模式链接上的难度，并以正确的 SQL 或 Cypher 风格生成相应位置的子句。在 Decoder 部分，本文采用了 teacher forcing 策略来训练模型，为其提供正确的 PQL 结构特征，以便正确地判断 SQL 和 Cypher 的嵌套位置。

具体来说，模型 $\mathcal{M}$ 预测 PQL 查询语句 $\mathcal{Y}$ 的概率分布，并以逐词的形式生成查询语句。预测的概率分布的形式表示如下：

$$P_{\mathcal{M}}(\mathcal{Y}, \mathcal{B}|\mathcal{Q}, \mathcal{S}, \mathcal{A}) = \prod_{i=1}^{|\mathcal{Y}|} P_{\mathcal{M}}(\mathcal{Y}_i, \mathcal{B}_i|\mathcal{Q}, \mathcal{S}, \mathcal{A}, \mathcal{Y}_j, j < i)$$

此处 $\mathcal{Y}_i$ 是 PQL 查询语句 $\mathcal{Y}$ 的某个前缀，$P_{\mathcal{M}}(\mathcal{Y}_i, \mathcal{B}_i|\cdot)$ 表示给定前缀 $\mathcal{Y}_j, j < i$，自然语言问题 $\mathcal{Q}$，数据库模式 $\mathcal{S}$ 和 scope 信息 $\mathcal{A}$ 后查询语句 $\mathcal{Y}$ 的第 i 个词和对应位置 skeleton 信息 $\mathcal{B}_i$ 的概率分布。

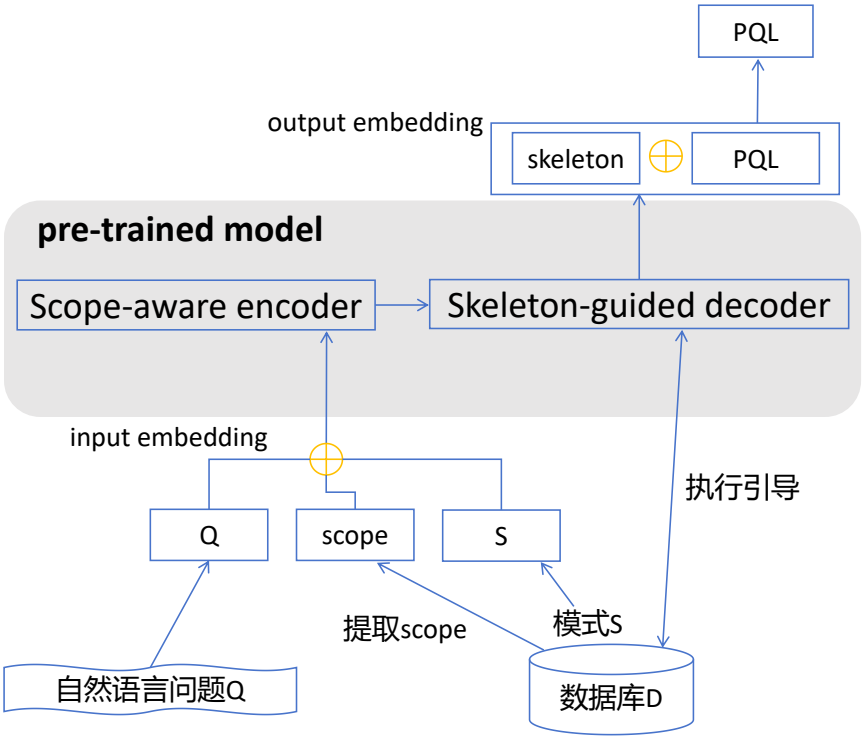

图 3.1　SESD 模型架构

3.2　模型细节

3.2.1　Scope-Aware Encoder

模式链接　只给模型提供自然语言问题，模型将很难从问题中提取出对其生成 PQL 有帮助的信息，一个重要的原因是其无法区别自然语言问题中表意的词和与数据库模型相关的词。因此本文将数据库模式信息和 scope 信息同自然语言问题一起作为输入序列 $X = q|sql : t_1, t_2, ...|cypher : ...|t_1 : c_1^1, ..., c_{n^1}^1|...|t_N : c_1^N, ..., c_{n^N}^N$，其中 | 是分隔符，$q$ 表示自然语言问题，t_i 表示数据库实体名称，c_i^j 表示实体 t_i 的第 j 个属性。为了使模型更好地学习，应该让查询语句中涉及到的变量尽可能出现在自然语言问题中，也就是说，数据库模式信息应该尽量是自然语言问题中的原词复现。

3.2.2　Skeleton-Guided Decoder

PQL 规范化　本文所使用到的数据集存在着关键字大小写混杂的情况。为方便模型的理解，本文将所有数据都转为小写的形式输入给模型。此外在 Advising[12]，Restaurants[13]，IMDB[31]三个数据集中，大量使用了 AS 关键字命名别名。使用别名后，模型在读取实体属性时可能会无法与数据库模式信息对应上，这将影

表 3.1　选自 IMDB 的 PQL 结构示例

自然语言问题 q	What are all the movies directed by ”Quentin Tarantino” featuring ”Christoph Waltz”
PQL	MATCH (MOVIEALIAS0 : MOVIE) - [DIRECTED_BYALIAS0 : DIRECTED_BY] -> (DIRECTEDALIAS0 : DIRECTOR) , { SELECT CASTALIAS0.msid FROM ACTOR AS ACTORALIAS0 , CAST AS CASTALIAS0 WHERE CASTALIAS0.aid = ACTORALIAS0.aid AND ACTORALIAS0.name = ”christoph waltz” } WHERE DIRECTEDALIAS0.name = ”quentin tarantino” AND MOVIEALIAS0.mid = CASTALIAS0.msid RETURN MOVIEALIAS0.title
规范化 PQL	match (movie : movie) - [directed_by : directed_by] -> (director : director) , { select cast.msid from actor as actor , cast as cast where cast.aid = actor.aid and actor.name = ”christoph waltz” } where director.name = ”quentin tarantino” and movie.mid = cast.msid return movie.title
PQL 结构	match _ { select _ from _ where _ } where _ return _

响模型学习的效果。因此本文将查询语句中别名的部分删除，简单使用实体名称和属性名称。表3.1的第二行是原始数据集的 PQL，包含如“ACTOR AS actoralias0”，“actoralias0.name = ‘hristoph waltz’”等语句。第三行是规范化后的 PQL，已经删除了别名的使用，并将关键字都转换为小写。

结构特征提取　受到 NL2SQL 任务上基于草图和槽位填充的方法[22-23,39-40]的启发，本文在 Decoder 部分添加了 PQL 的结构特征，引导模型生成 PQL。对于一个 PQL 查询，本文使用正则匹配的方法匹配其中的 PQL 关键字，如“select”、“match”、“where”、“{”等，保留关键字的部分并将其余部分替换为下划线（_）。值得注意的是，有一些关键字被本文忽略了，如“distinct”、“and”、“group by”等，因为并不是所有查询都具备这些关键词，并且从自然语言问题中也难以提取出对应的信息。表3.1第四行显示了由规范化后的 PQL 提取出的结构特征。

执行引导策略　由于 SESD 的 Decoder 以序列的方式生成 PQL，因此模型很可能会生成不符合语法的 PQL。为了缓解这一问题带来的影响，本文使用执行引导（execution guidance）策略[41]，在解码过程中使用束搜索（beam search），选择第一个可执行的 PQL 作为最终的输出。如图3.2所示，解码过程中模型生成了多个

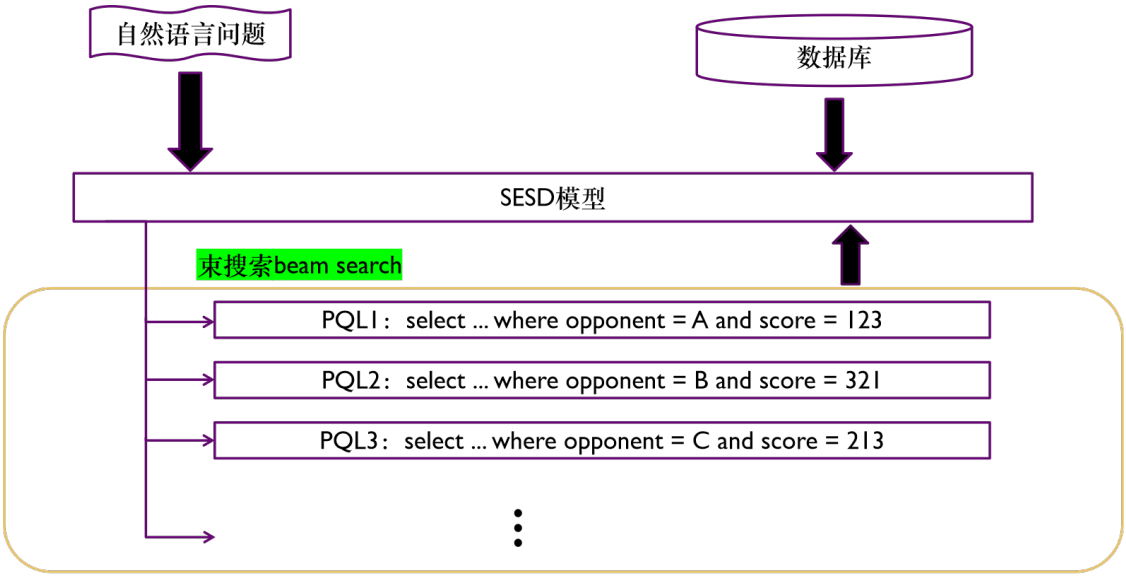

图 3.2　SESD 使用的执行引导策略的工作流程

可能的 PQL，依次执行它们并选择了首个能成功执行的 PQL。虽然加入执行引导的解码器仍有可能生成与真实结果不符的查询语句，但已经排除了许多无法执行的错误结果。

3.3　翻译方法流程

为进一步阐明本文提出的面向多类型查询语句的类型和结构信息的翻译方法，本节将通过具体的例子说明 SESD 模型的工作流程。

首先，给定自然语言问题“Find the actor who played 'Captain Miller' in the movie 'Saving Private Ryan'”，我们将其对应的数据库模式信息和 scope 信息合并，以字符“|”分割，得到形式如“自然语言问题 | scope 信息 | 数据库模式信息”的字符串：“Find the actor who played 'Captain Miller' in the movie 'Saving Private Ryan' | sql: actor, cast, ... | cypher: director, movie, ... | actor: actor.aid, actor.gender, ... | cast: cast.id, ... | ... | director: director.did, director.gender, ... | ...”。上述字符串在词嵌入后被输入进模型的 Encoder 部分。自然语言问题对应的 PQL 和 PQL 结构被合并作为 Decoder 部分的输出，同样以字符“|”分割，得到形式如“PQL 结构 | PQL”的字符串：“select _ from _ match _ return _ where _ | select actor.name from ... match ... return ... where ...”。由于采用 teacher forcing 策略，因此训练时会将输出字符串作为 Decoder 部分的输入，帮助模型快速收敛。

第 4 章　实验设计与结果

本章首先介绍了实验的基本设置，包括实验的环境，使用的数据集，基线方法和评估指标，然后介绍了实验数据预处理的内容，接着介绍了 SESD 的实现细节，最后介绍了对比实验和消融实验的实验结果，并对几个正确和错误的案例进行了分析。

4.1　实验设置

4.1.1　实验环境

所有的实验都在同一台服务器上完成，服务器配置如下：

- CPU：Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz
- GPU：NVIDIA GeForce RTX 3090
- 内存：256G
- 操作系统：Ubuntu 20.04.5 LTS

4.1.2　数据集

本文在2.2节提到的四个数据集上进行了广泛的实验，以评估模型在不同分布的数据上的表现效果。对于规模庞大的数据集（WikiSQL、Advising），本文将数据以 7：1：2 的比例划分训练集、验证集和测试集。对于规模较小的数据集（Restaurants、IMDB），本文使用 k 折交叉验证（k-fold cross-validation）的方式划分数据集。实验中，本文将 k 设置为 5，随机划分数据集。

4.1.3　基线方法

除了2.3节提到的三种方法：Seq2seq[32]、Seq2seq+attention[32]和 Seq2SQL[15]，本文还使用了预训练模型 BART[42]、T5[25]和 Rothe 等人设计的 BERT2BERT[43]作为基线方法。

4.1.4　评估指标

多年来，深度学习社区和自然语言处理社区都为自然语言处理及其下游任务提出了大量的评估指标。针对机器翻译任务，常用指标有 BLEU[44]、METEOR[45]、ROUGE[46]、TER[47]、chrF[48]等。

BLEU BLEU[44]是机器翻译领域最古老、最常用、最经典的指标之一，许多机器翻译论文甚至只使用这一个指标作为系统的评价标准，其重要性不言而喻。BLEU[44]方法的核心思想是：机器翻译系统的输出与专业人工翻译的结果越接近，系统的质量就越高。这里“接近”程度是通过计算翻译结果中词语和短语对人工答案的命中度来评价的，也就是说 BLEU[44]实质上是对翻译结果准确率的度量。具体而言，BLEU[44]计算机器翻译结果与真实结果之间的 n 元重叠率。通过计算不同长度的 n 元重叠率的几何平均数，得到 BLEU 分数。BLEU 分数还考虑了机器翻译结果与真实结果在长度上的匹配度，通过惩罚机制降低长度差异对分数的影响。BLEU 分数范围在 0 到 1 之间，分数越高表示翻译质量越好。BLEU[44]的优点在于计算简单，但它只关注翻译结果与真实结果之间的字面匹配，因此可能忽视了语法和语义上的准确性。

METEOR METEOR[45]是一种综合性的翻译评价指标，考虑了精确匹配、词形变化匹配、同义词匹配以及词序等多个方面。具体来说，METEOR[45]计算机器翻译结果与真实结果之间的精确匹配，包括单词匹配和短语匹配，考虑词形变化和同义词匹配，提高对语言多样性的容忍度，通过计算精确匹配、词形变化匹配和同义词匹配的加权得分，得到最终分数，并根据机器翻译结果与真实结果之间的词序差异，进行惩罚或加分。METEOR[45]同时考虑了精确度和召回率，而不是仅仅关注其中之一，这提供一个更全面的评估。同时 METEOR[45]考虑了词序，这是许多其他指标都忽略的。但 METEOR[45]仍然可能会忽略一些语义或句法差异，尤其是对于长序列。

ROUGE ROUGE[46]是主要基于召回率的评估指标，常用的有四种变体指标。ROUGE-N 统计 n 元序列的召回率。ROUGE-L 考虑了机器翻译结果和真实结果之间的最长公共子序列。ROUGE-W 改进了 ROUGE-L，用加权的方法计算最长公共子序列。ROUGE-S 也是统计 n 元序列，但是其允许跳词，即单词不需要连续出现。ROUGE[46]的计算十分高效，但一个重大的缺点是只能从单词、短语的角度去衡量两个句子的形似度，会忽略同义词、近义词等语义信息。

TER TER[47]通过计算将机器翻译结果转换为真实结果所需的编辑操作次数来评估翻译质量。编辑操作包括四项插入、删除、单词替换和平移四种，每个操作代价都为 1。由于考虑平移操作的编辑距离计算是 NP 完全的，因此 TER[47]退而求其次，结合了贪心算法和动态规划算法。先贪心地寻找（伪）最优的平移操作，

在结束所有的平移操作后，使用动态规划算法来求解除平移外的编辑距离。利用编辑次数来衡量翻译系统的表现效果十分直观且细致，但也可能会导致过于严格的评估。同时编辑次数不考虑语义正确但措辞不同的表达，可能忽略同义词或词形变化的匹配。

chrF　chrF[48]与其他几种评估指标最大的不同在于它是字符级的。chrF[48]是一种基于字符的 n 元翻译评估指标，它考虑了一些形态——句法现象，除此之外，与其他评估指标相比，它很简单，不需要任何额外的工具或知识来源，它完全独立于语言，也独立于分词过程。chrF[48]提供了更细粒度的翻译评估，它可以更好地捕捉机器翻译结果与真实结果之间的相似性，特别是对形态丰富的语言，但对长文本中的语义和上下文关系的评估上效果不佳。

上述的几种评估指标在面对自然语言时表现良好，但并不适于语法特殊、结构复杂的数据库查询语言。因为查询语言并不像自然语言一样具备直观的语义，查询语言和自然语言之间也不具备单词或字符之间的对应关系。更重要的是，查询语言应该具备在指定数据库系统执行的能力，这是自然语言无需考虑的标准。

为了评估 NL2PQL 模型的表现效果，本文采用了两个指标：逻辑形式准确率（logic form accuracy，LF）[40]和执行准确率（execution accuracy，EX）[16,49]。这是 NL2SQL 任务中最常使用的两个评估指标。逻辑形式准确率衡量生成的 PQL 是否与真实 PQL 在字面上完全一致，执行准确率衡量执行生成的 PQL 得到的结果是否与真实的结果完全一致。通常来说，执行准确率会略高于逻辑形式准确率，因为执行准确率允许条件顺序的差异造成逻辑形式的不一致。

4.2　数据预处理

数据规范化　根据3.2.2节的规范化方法，本文将数据集中的 PQL 规范化后输入进模型。

数据划分　WikiSQL[15]原始数据集已划分好训练集、验证集和测试集，本文按照4.1.2节提到划分方法对其余三个数据集进行划分。

表 4.1　基线方法和 SESD 在 4 个数据集上的 LF 和 EX（%）

数据集 / 方法	WikiSQL		Advising		Restaurants		IMDB	
	LF	EX	LF	EX	LF	EX	LF	EX
Seq2seq	1.1	1.3	23.4	26.6	8.0	8.0	3.2	3.2
Seq2seq+attention	1.2	1.3	27.6	32.4	9.6	9.6	6.9	6.9
Seq2SQL	45.6	56.0	N/A	N/A	N/A	N/A	N/A	N/A
BART-large	44.6	61.4	6.2	8.2	0.0	3.0	0.0	0.0
BERT2BERT	N/A	N/A	0.0	0.0	0.0	0.0	0.0	0.0
T5-base	46.2	61.4	83.3	89.4	44.0	60.0	32.2	32.2
SESD-large(ours)	**70.6**	**81.4**	**88.0**	**92.5**	**100.0**	**100.0**	**58.6**	**66.2**
SESD-base(ours)	70.3	81.2	86.7	91.2	52.0	72.0	41.3	43.5

4.3　SESD 实现细节

SESD 模型基于预训练模型 T5[25]微调得到。针对不同的数据集和预训练模型，本文选取了不同的批大小（batch size）和学习率（learning rate）。本文采用 Adafactor[50]优化器进行训练，训练过程中，使用预热（linear warm-up）和衰减（cosine decay）的方式动态调整学习率，加快收敛速度，减少过拟合风险。在束搜索（beam search）阶段，本文将搜索分支设为 8，以排除一些存在语法错误的 PQL。

4.4　准确率实验结果

在 NL2SQL 任务上，通常逻辑形式准确率和执行准确率会有一定差别，执行准确率略高于逻辑形式准确率。但从表4.1中可以观察到，Seq2seq 模型在四个数据集上的逻辑形式准确率（LF）和执行准确率（EX）基本一致。这是因为 SQL 和 Cypher 的互相嵌套导致 PQL 的结构远比 SQL 复杂，预测的 PQL 出现语法错误的形式更多、概率更大了。并且原先集中在 SQL 的 WHERE 子句部分的查询条件有一部分被移至了 Cypher 的 WHERE 子句部分，因此，由于查询条件顺序的差异所造成的逻辑形式的错误不再那么容易出现。

此外，本文发现增加了注意力机制的 Seq2seq 模型在表现效果上并没有得到显著提升。这是因为 Seq2seq 模型简单的 Encoder-Decoder 结构过于通用化，难以

理解在形式和结构上相较于自然语言更为复杂的查询语言的语义。因此，该模型往往无法生成有效的查询语言。

Seq2seq 模型在 Advising[12]数据集上表现相对较好，而在 WikiSQL[15]数据集、Restaurants[13]数据集和 IMDB[31]数据集上表现相对较差，这主要是数据集的难度不同所致。WikiSQL[15]数据集的 PQL 形式差异较大，且涉及多个数据库，训练集、验证集和测试集也经由专门的团队进行划分，因此模型不仅需要学习“理解”训练集的数据，还需要学会“推广”到未见过的数据库上。相反，另外三个数据集只有一个数据库，这意味着，从某种程度上说，测试集的数据库模式信息对模型来说已经是见过的，因此难度较 WikiSQL[15]数据集要稍微低一些。Advising[12]数据集规模适中，查询语句结构差异较大，因此基线方法的正确率相对较低。IMDB[31]数据集虽然规模较小，但其中 PQL 的结构最为复杂，且自然语言问题与查询语句几乎是一一对应的，这意味着结构相似的查询语句很少，因此，模型在学习 IMDB 数据集时的表现较差。而 Restaurants[13]数据集虽然规模最小，但数据库也仅有 3 个，模型需要学习的内容实际上有限，因此模型的准确率略高于 IMDB[31]数据集。

采用了基于槽位填充思想的 Seq2SQL 模型，受查询语句结构的限制较大，只能接受 WikiSQL[15]数据集中的查询语句的结构。因此本文只在 WikiSQL[15]数据集上训练并测试了适配后的 Seq2SQL 模型。从表4.1的结果能看到，Seq2SQL 的表现明显优于 Seq2seq，这也符合本文的预期。因为 Seq2SQL 在输入查询语句的不同子句上添加了模块来进行专门预测，增强了模型对查询语句结构的感知能力，从而避免了模型预测大量语法错误的查询语句。

BART[42]是序列生成任务常用的预训练模型，但在 NL2PQL 任务上表现不佳。观察生成的 PQL，可以发现模型能够生成符合语法的 PQL，但查询条件和属性名会出错。这可能与 BART[42]预训练任务相关。BART[42]预训练任务是恢复被破坏的文本，可能导致在 NL2PQL 任务上 BART[42]错误理解了翻译目的。

BERT2BERT[43]在 NL2PQL 任务上完全失败了，在所有数据集上的逻辑形式准确率和执行准确率都是 0。实验发现 BERT2BERT[43]在训练集上能较快收敛，但测试集上甚至无法生成符合语法的 PQL。造成这种结果的原因一方面可能是 BERT 本身并不擅长生成任务，另一方面可能是构造的数据集规模太小，模型无法充分学习 PQL 特征。

T5[25]将所有任务都转化为文本到文本，本身就是机器翻译领域常用的预训练模型，在 NL2PQL 任务的几个基线方法中表现效果最好。

表 4.2　SESD-base 移除类型和结构信息的准确率（%）

数据集 / 模型变量	WikiSQL		Advising		IMDB	
	LF	EX	LF	EX	LF	EX
SESD-base	70.3	81.2	86.7	91.2	41.3	43.5
- w/o skeleton	70.0	80.6	85.8	90.6	37.9	39.4
- w/o scope	46.7	62.6	83.5	89.8	34.4	34.4

根据使用的预训练模型的参数规模，本文实现了 SESD 的两个版本：SESD-base 和 SESD-large。base 版本拥有 2.2 亿左右个参数，large 版本拥有 7.7 亿左右个参数。

表4.1展示了 SESD 在四个数据集上的执行结果。相较于基线方法，SESD 的逻辑形式准确率和执行准确率都取得了显著的提升。与基线方法中最优的 T5-base[25]相比，SESD-large 在 WikiSQL[15]数据集上逻辑形式准确率提升了 24.4%，执行准确率提升了 20%。在 Advising[12]数据集上，逻辑形式准确率提升了 4.7%，执行准确率提升了 3.1%。在 IMDB[31]数据集上，逻辑形式准确率提升了 26.4%，执行准确率提升了 34%。在 Restaurants[13]数据集上，SESD-large 的逻辑形式准确率和执行准确率都达到了 100%。这是因为 Restaurants[13]原始数据集过于简单，只涉及一个数据库、三个实体，且数据集规模小，数据预处理阶段随机的交叉验证划分可能导致训练集和测试集的分布不独立，且与原数据集的分布不同。在前几轮训练中模型预测的结果尚不准确，但经过几十轮的训练后就能达到 100% 的准确率。

4.5　消融实验结果

为了验证 SESD 的 Encoder 和 Decoder 都提升了模型的效果，本文设计了消融实验（ablation study）。

本文使用 SESD-base 模型在 WikiSQL[15]、Advising[12]和 IMDB[31]数据集上进行了全面的消融实验，以分析各个组件对模型性能的影响。实验结果如表4.2所示。

从结果可以看出，当移除 skeleton 信息后，逻辑形式准确率和执行准确率均平均下降了 1%-2%。这是因为自然语言和查询语言在语法和句型结构方面存在较大差异。使用 PQL 的结构作为中间解释层有助于模型更好地理解 PQL。

移除 scope 信息后，SESD-base 在 Advising[12] 数据集和 IMDB[31] 数据集上逻辑形式准确率和执行准确率均平均下降了 3%-4%。而在 WikiSQL[15] 数据集上，逻辑形式准确率下降了 23.3%，执行准确率下降了 18%。这表明在处理涉及多个模式的 PQL 时，scope 信息对于帮助模型理解数据库模式起着重要作用。尤其对于难度较高的数据集 WikiSQL[15]，由于测试集的数据库信息对于训练集是完全不可见的，因此模型无法在训练的时候“记住”数据库信息，必须真正“学会”如何理解 PQL。

4.6 案例分析

本节对 SESD 和其他基线方法生成的 PQL 进行了分析。我们首先关注了几个基线方法翻译错误但 SESD 翻译正确的案例，然后讨论了几个 SESD 翻译错误的案例。

4.6.1 正确案例

案例 1

自然语言问题：How many movies about Iraq war were produced in 2015 ?

真实 PQL：

```
match ( movie : movie ) , {
    select tags.msid
    from keyword , tags
    where keyword.keyword = "Iraq war"
    and tags.kid = keyword.id
}
where movie.release_year = 2015
and tags.msid = movie.mid
return count( distinct ( movie.title ) )
```

基线方法在预测这个复杂的 PQL 时全都失败了。对于跨模型的嵌套查询，基线方法完全无法理解该如何连接多个实体，生成的 PQL 存在大量语法错误，而 SESD 则能正确理解跨模型的需求。

Seq2seq+attention 预测 PQL：

```
match ( movie : movie ) , tags.msid
from keyword , tags
```

```
where keyword.keyword = "Iraq war"
}}
where tags.msid = movie.mid
select count( distinct ( movie.title ) )
```

案例 2

自然语言问题：Can I take EHS 642 as MDE ?

真实 PQL：

```
select count( * ) > 0
from course , program , program_course
where course.department = "EHS"
and course.number = 642
and program_course.category like "%MDE%"
and program_course.course_id = coursea.course_id
and program.name like "%cs-lsa%"
and program.program_id = program_course.program_id
```

对于不涉及跨模型的 PQL，基线方法能够基本预测正确。然而，在自然语言问题稍微复杂的情况下，基线方法很难从语义中理解聚合的概念，并正确使用聚合函数。例如，当自然语言问题中显式地使用“how many”提问时，模型能理解应该使用 count 函数，但在隐式地提问时，模型就开始胡乱预测了。这种情况下，SESD 仍然能够正确预测。

Seq2seq+attention 预测 PQL：

```
select program.name, program_course.number
from course , program , program_course
where course.department = "EHS"
and course.number = 642
and program_course.category like "%MDE%"
and program_course.course_id = coursea.course_id
and program.name like "%cs-lsa%"
and program.program_id = program_course.program_id
```

4.6.2 错误案例

案例 3

自然语言问题：What is the nationality of ”Kevin Spacey” ?

真实 PQL：

```
select writer.nationality
from writer
where writer.name = "Kevin Spacey"
```

预测 PQL：

```
match ( director:director )
where director.name = "Kevin Spacey"
return director.nationality
```

实体 Writer 和 Director 都有属性 nationality 和 name，但自然语言问题中没有显式指明”Kevin Spacey” 是 Writer 还是 Director。对于人类，可以通过额外知识确定应该查询 Writer 还是 Director，但模型无法判断使用哪个实体，因此错误地预测了 PQL。

案例 4

自然语言问题：Who are the instructors of the hardware courses?

真实 PQL：

```
select distinct instructor.name
from instructor
...
join { match (area:area) return area }
on course_offering.course_id = area.course_id
where area.area like "%hardware%"
```

预测 PQL：

```
select distinct instructor.name
from instructor
...
join  match (area:area) return area
on course_offering.course_id = area.course_id
where area.area like "%hardware%"
```

可以观察到，模型在基本预测了 PQL 的情况下，未能正确预测大括号 ()，而是将大括号预测为了空格。经过多次实验，本文发现增加训练轮数可以缓解甚至解决这种错误。对于模型来说，可能难以理解 PQL 中大括号这种与语义无关的字符。

案例 5

自然语言问题：What is the theme for the episode selection process ?

真实 PQL：

```
match (n:table)
where n.episode = selection process
return n.theme
```

预测 PQL：

```
match (n:table)
where n.result = select n.theme
from table
where episode = selection process
```

可以发现，预测的 PQL 实际上包含了真实 PQL 的内容，但错误地混杂了 SQL 和 Cypher 的语法，并添加了不相关的内容。“select”是 PQL 语法中的关键词，而自然语言问题中出现了“select”的变体“selection”，模型似乎无法明确此处的“select”意义。虽然模型成功地以 Cypher 风格预测了该实体，但受到“select”影响，又预测了一个 SQL 的伪嵌套。

4.7 实验总结

本章进行了大量实验，结果表明，本文提出的 SESD 在四个数据集上都显著优于基线方法，逻辑形式准确率和执行准确率均大幅提升，展现出强大的 NL2PQL 翻译能力。其中，SESD-large 表现略优于 SESD-base。消融实验证明了 SESD 的 Encoder 和 Decoder 的设计都提高了模型在处理 NL2PQL 问题上的能力。然而，在实验过程中也发现当前数据集规模和难度有限，无法充分评估模型效果，需要更大规模和更高难度的数据集进行进一步评估。

第 5 章　系统实现

本章主要介绍本研究中实现的自然语言到多类型查询语言的翻译的可视化系统，支持用户便捷地在本文提出的四个数据集上，输入自然语言问题，得到模型翻译的 PQL 和对应的查询结果。

5.1　系统架构

系统按图5.1组织成三个模块：用户交互模块，管理模块，NL2PQL 模型集。数据库提前内置在系统中。

用户交互模块　用户交互模块提供用户交互的功能，图5.2展示了可供交互的页面。用户可选择某个数据集，并输入自然语言问题，用户交互模块将信息打包发送给管理模块。同时，用户交互模块会接收来自管理模块的信息，展示 PQL 和查询结果。

管理模块　管理模块负责整个系统的数据调度。在接收来自用户交互模块的自然语言问题和数据集信息后，管理模块会连接数据库管理系统，寻找对应的数据库模式信息，并将其与自然语言问题按照第3章所述方式进行编码，得到 input embedding。然后根据所选的数据集在 NL2PQL 模型集中选出一个对应的训练好的模型，将 input embedding 作为输入，得到模型生成的 output embedding。管理模块能够从 output embedding 中解码得到 PQL，并连接数据库管理系统进行查询，得到查询结果。管理模块还将负责将 PQL 和查询结果发送给用户交互模块。

NL2PQL 模型集　NL2PQL 模型集包括了在四个训练集上训练好的 SESD-large 模型，负责接收编码好的自然语言问题，生成 PQL，是整个系统的核心部分。

整个系统的工作流程如下：用户在前端页面选择四个数据集中的任意一个，并输入自然语言问题，通过表单提交的方式将参数传到后端后，后端将自然语言问题输入对应训练好的模型，生成 PQL，并交由数据库执行，最终将 PQL 和执行结果一同返回前端页面展示。

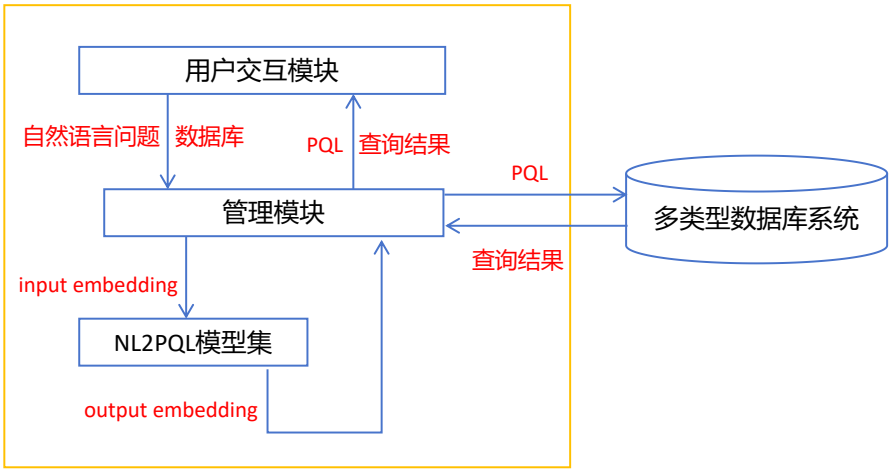

图 5.1　自然语言到多类型查询语言的翻译的可视化系统架构

图 5.2　自然语言到多类型查询语言的翻译的可视化系统页面展示

5.2　技术选型

本文实现的可视化系统采用前后端分离的架构，前端使用 Vue.js 框架，后端使用 Flask 框架，数据库使用 SQLite，模型推理使用 PyTorch 框架，多类型数据库系统基于 MySQL 和 Neo4J 搭建。

前后端接口设计 系统的前后端主要依赖一个接口”predict”，具体设计如下：

- 请求方式：POST
- 请求参数：
  - dataset: 数据集名称，取值为”wikisql”、”imdb”、”restaurant”、”advising”
  - question: 自然语言问题
- 返回参数：
  - pql: 生成的 PQL
  - result: 查询结果

数据库设计 系统的数据库主要存储记录用户查询请求的日志，包括用户查询的数据集、自然语言问题、生成的 PQL、查询结果等信息。数据库表设计如下：

- log
  - id: 主键，自增
  - dataset: 数据集名称
  - question: 自然语言问题
  - pql: 生成的 PQL
  - result: 查询结果
  - created_time: 创建时间

多类型数据库系统设计 与系统交互的 polystore 系统包含了关系数据库 MySQL 和图数据库 Neo4J。根据输入的 PQL，系统会根据实体的类型选择合适的数据库进行查询。对于关系型数据集，系统会使用 MySQL 进行查询，对于图数据集，系统会使用 Neo4J 进行查询。

算法 5.1 查询多类型数据库系统

Input: PQL
Output: 查询结果
1: 分离 PQL 中 SQL 和 Cypher 部分
2: **if** PQL 中包含 SQL 部分 **then**
3: 使用 MySQL 执行 SQL 部分
4: **end if**
5: **if** PQL 中包含 Cypher 部分 **then**
6: 使用 Neo4J 执行 Cypher 部分
7: **end if**
8: 将结果通过外键合并成表的形式

第 6 章　总结与展望

本文创新地提出了 NL2PQL 这一全新的自然语言处理领域的任务，并为其提供了对应的基准数据集和基线方法。同时，本文深入探究了深度学习技术，特别是预训练技术在解决该任务上的应用和可行方法。大量的对比实验和分析证实了本文提出的针对 PQL 的设计在所提出的 4 个数据集上都表现出了十分可观的自然语言到多类型查询语言的翻译能力。

然而，第 4 章的实验结果反映了一个明显的不足：缺乏规模庞大且足够难度的数据集。受到时间和人力资源的限制，本文提出的 4 个数据集或者规模过小，或者难度过低，或者两者兼而有之。现有 NL2PQL 的数据集难度已经无法对模型的翻译能力进行全面而细致的评估。因此，提出一个规模合适、难度足够的数据集应成为后续研究的重点，这也是一个不可避免的关键点。本文使用的参数规模最大的模型未达到 1B 的规模，后续研究可以尝试使用 7B、13B 或者更大规模的预训练模型进行微调。值得一提的是，本文尝试使用大模型 prompt 工程的方法解决 NL2PQL 任务，但是由于大模型没有接受过 PQL 的数据训练，无法很好的理解的 PQL 的语法和语义，因此效果并不理想。此外，本文使用的 PQL 只涉及了关系型和图两种类型的数据模型，但实际工程中往往还需要如文档、键值类型的数据模型，因此后续工作可以尝试考虑更多的数据模型。

未来 NL2PQL 任务还有着许多发展方向和潜在应用：

1. 多模态支持。可以尝试将自然语言扩展到其他模态，如图像、视频或音频，从而多样化地获取查询语言。这将为用户提供更直观、更全面的查询方式，进一步提升系统的鲁棒性和可用性。
2. 跨语言支持。随着全球化的不断深入，跨语言查询将成为一个重要的需求。未来研究可以致力于实现跨语言的 NL2PQL 转换，自然语言不再囿于英语，用户可以使用自己的母语进行查询。
3. 实时交互和反馈机制。可以为 NL2PQL 系统引入实时交互和反馈机制，使得用户可以在查询过程中进行实时调整和反馈，系统可以根据用户的反馈实时调整查询语言的生成过程，从而更好地满足用户的需求和意图。
4. 领域特定查询。针对特定领域的查询需求，可以设计定制化的 NL2PQL 系统，使得系统能够更好地理解和处理特定领域的查询语言，这将为特定领域的用户提供更加高效、精准的查询服务。

在大数据时代，polystore 系统将扮演着至关重要的角色。这种综合多个数据存储和管理系统的方法不仅能够有效地整合和管理庞大的数据资源，还能够提供更加灵活和高效的数据访问和处理方式。随着数据量的不断增长和数据类型的不断多样化，polystore 系统的研究和应用将成为研究人员们的一个重要议题。

在工业生产领域，polystore 系统也将得到越来越广泛的应用。在制造业、物流业、金融业等领域，企业需要处理大量的数据，并且需要从这些数据中提取有价值的信息和洞见。Polystore 系统的引入可以帮助企业更好地整合和利用分布在不同数据源中的数据，从而提升数据处理和分析的效率和准确性，加速决策过程，提高企业的竞争力和创新能力。

NL2PQL 任务的研究对于 polystore 系统的发展具有重要意义。通过将自然语言转换为多类型查询语言，NL2PQL 任务使得研发人员和非专业的数据库使用者能够更加方便快捷地上手和使用 polystore 系统。这将降低系统的门槛，扩大系统的受众群体，促进 polystore 系统在各个领域的广泛应用和发展。

总体来说，NL2PQL 任务的研究不仅有助于提升 polystore 系统的普适性和可用性，还能够推动工业生产和全行业的进步和发展。随着技术的不断进步和研究的不断深入，相信 polystore 系统将在未来发挥越来越重要的作用，为社会带来更多的便利和发展机遇。

插图索引

表格索引

参考文献

[1] BROWN P G. Overview of scidb: large scale array storage, processing and analysis[C]// Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. 2010: 963-968.

[2] STONEBRAKER M, ÇETINTEMEL U. ” one size fits all” an idea whose time has come and gone[M]//Making databases work: the pragmatic wisdom of Michael Stonebraker. 2018: 441-462.

[3] DUGGAN J, ELMORE A J, STONEBRAKER M, et al. The bigdawg polystore system[J]. ACM Sigmod Record, 2015, 44(2): 11-16.

[4] VOGT M, STIEMER A, SCHULDT H. Polypheny-db: towards a distributed and self-adaptive polystore[C]//2018 IEEE International Conference on Big Data (Big Data). IEEE, 2018: 3364-3373.

[5] BAIK C, JAGADISH H V, LI Y. Bridging the semantic gap with sql query logs in natural language interfaces to databases[C]//2019 IEEE 35th International Conference on Data Engineering (ICDE). IEEE, 2019: 374-385.

[6] LI F, JAGADISH H V. Constructing an interactive natural language interface for relational databases[J]. Proceedings of the VLDB Endowment, 2014, 8(1): 73-84.

[7] SAHA D, FLORATOU A, SANKARANARAYANAN K, et al. Athena: an ontology-driven system for natural language querying over relational data stores[J]. Proceedings of the VLDB Endowment, 2016, 9(12): 1209-1220.

[8] IYER S, KONSTAS I, CHEUNG A, et al. Learning a neural semantic parser from user feedback[A]. 2017.

[9] 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. 2003: 149-157.

[10] IYER S, KONSTAS I, CHEUNG A, et al. Learning a neural semantic parser from user feedback[A]. 2017.

[11] LI F, JAGADISH H V. Constructing an interactive natural language interface for relational databases[J]. Proceedings of the VLDB Endowment, 2014, 8(1): 73-84.

[12] FINEGAN-DOLLAK C, KUMMERFELD J K, ZHANG L, et al. Improving text-to-sql evaluation methodology[A]. 2018.

[13] 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. 2003: 149-157.

[14] KIM H, SO B H, HAN W S, et al. Natural language to sql: Where are we today?[J]. Proceedings of the VLDB Endowment, 2020, 13(10): 1737-1750.

[15] ZHONG V, XIONG C, SOCHER R. Seq2sql: Generating structured queries from natural language using reinforcement learning[A]. 2017.

[16] 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[A]. 2018.

[17] SCHOLAK T, SCHUCHER N, BAHDANAU D. Picard: Parsing incrementally for constrained auto-regressive decoding from language models[A]. 2021.

[18] SHI P, NG P, WANG Z, et al. Learning contextual representations for semantic parsing with generation-augmented pre-training[C]//Proceedings of the AAAI Conference on Artificial Intelligence: volume 35. 2021: 13806-13814.

[19] SHAW P, CHANG M W, PASUPAT P, et al. Compositional generalization and natural language variation: Can a semantic parsing approach handle both?[A]. 2020.

[20] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[21] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[22] XU X, LIU C, SONG D. Sqlnet: Generating structured queries from natural language without reinforcement learning[A]. 2017.

[23] YU T, LI Z, ZHANG Z, et al. Typesql: Knowledge-based type-aware neural text-to-sql generation[A]. 2018.

[24] DEVLIN J, CHANG M W, LEE K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[A]. 2018.

[25] RAFFEL C, SHAZEER N, ROBERTS A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer[J]. Journal of machine learning research, 2020, 21(140): 1-67.

[26] HUI B, GENG R, WANG L, et al. S ^2 sql: Injecting syntax to question-schema interaction graph encoder for text-to-sql parsers[A]. 2022.

[27] GUO A, LI X, XIAO G, et al. Spcql: A semantic parsing dataset for converting natural language into cypher[C]//Proceedings of the 31st ACM International Conference on Information & Knowledge Management. 2022: 3973-3977.

[28] ZHANG L, CHAI C, ZHOU X, et al. Learnedsqlgen: Constraint-aware sql generation using reinforcement learning[C]//Proceedings of the 2022 International Conference on Management of Data. 2022: 945-958.

[29] BA J, RIGGER M. Testing database engines via query plan guidance[C]//The 45th International Conference on Software Engineering (ICSE'23). 2023.

[30] TANG X, WU S, ZHANG D, et al. Detecting logic bugs of join optimizations in dbms[J]. Proceedings of the ACM on Management of Data, 2023, 1(1): 1-26.

[31] YAGHMAZADEH N, WANG Y, DILLIG I, et al. Sqlizer: query synthesis from natural language[J]. Proceedings of the ACM on Programming Languages, 2017, 1(OOPSLA): 1-26.

[32] DONG L, LAPATA M. Language to logical form with neural attention[A]. 2016.

[33] CHUNG J, GULCEHRE C, CHO K, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling[A]. 2014.

[34] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient estimation of word representations in vector space[A]. 2013.

[35] PENNINGTON J, SOCHER R, MANNING C D. Glove: Global vectors for word representation[C]//Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014: 1532-1543.

[36] RADFORD A, NARASIMHAN K, SALIMANS T, et al. Improving language understanding by generative pre-training[M]. OpenAI, 2018.

[37] FLORIDI L, CHIRIATTI M. Gpt-3: Its nature, scope, limits, and consequences[J]. Minds and Machines, 2020, 30: 681-694.

[38] YU T, WU C S, LIN X V, et al. Grappa: Grammar-augmented pre-training for table semantic parsing[A]. 2020.

[39] LI H, ZHANG J, LI C, et al. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql[C]//Proceedings of the AAAI Conference on Artificial Intelligence: volume 37. 2023: 13067-13075.

[40] FU H, LIU C, WU B, et al. Catsql: Towards real world natural language to sql applications [J]. Proceedings of the VLDB Endowment, 2023, 16(6): 1534-1547.

[41] WANG C, TATWAWADI K, BROCKSCHMIDT M, et al. Robust text-to-sql generation with execution-guided decoding[A]. 2018.

[42] LEWIS M, LIU Y, GOYAL N, et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension[A]. 2019.

[43] ROTHE S, NARAYAN S, SEVERYN A. Leveraging pre-trained checkpoints for sequence generation tasks[J]. Transactions of the Association for Computational Linguistics, 2020, 8: 264-280.

[44] PAPINENI K, ROUKOS S, WARD T, et al. Bleu: a method for automatic evaluation of machine translation[C]//Proceedings of the 40th annual meeting of the Association for Computational Linguistics. 2002: 311-318.

[45] BANERJEE S, LAVIE A. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments[C]//Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization. 2005: 65-72.

[46] LIN C Y. Rouge: A package for automatic evaluation of summaries[C]//Text summarization branches out. 2004: 74-81.

[47] SNOVER M, DORR B, SCHWARTZ R, et al. A study of translation edit rate with targeted human annotation[C]//Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers. 2006: 223-231.

[48] POPOVIĆ M. chrf: character n-gram f-score for automatic mt evaluation[C]//Proceedings of the tenth workshop on statistical machine translation. 2015: 392-395.

[49] ZHONG R, YU T, KLEIN D. Semantic evaluation for text-to-sql with distilled test suites[A]. 2020.

[50] SHAZEER N, STERN M. Adafactor: Adaptive learning rates with sublinear memory cost [C]//International Conference on Machine Learning. PMLR, 2018: 4596-4604.

致　　谢

衷心感谢王朝坤教授和石耕源学长的悉心指导和耐心指教。从选题到实验再到论文写作，他们都给予了我很多宝贵的建议和意见，让我开始初窥科研门径。王老师和耕源学长严谨的科研态度和认真的钻研精神深深感染了我，让我受益匪浅。感谢实验室的学长学姐们每周组会的分享和讨论，让我了解更多的知识和技术，眼界更加开阔，思维更加活跃。

感谢本科期间所有的授课老师和助教们，是你们让我在专业上不断深入，让我对计算机科学有了更深的理解，在本科四年的学习生涯中不断进步。感谢你们的辛勤付出和耐心教导。

感谢我的家人，是你们在背后默默支持着我，让我在最低谷的时候也能感受到家庭的温暖。感谢我的父母，是你们在我迷茫的时候给予我鼓励和支持，让我不断前行。你们用自己的人生经历为我树立了最好的榜样，值得我一生去学习和追随。

感谢本科四年的同学和朋友，是你们陪伴我度过了这段难忘的时光。感谢我的室友们，每天的相互陪伴和课程上的互相帮助推动着我不断进步。感谢班主任王继良老师和辅导员游凯超学长、吴海旭学长四年来在学习生活上对我的关照。

感谢初中高中的同学和朋友，大家一起度过了青春的时光，在毕业之后仍保持着亲密的联系，是你们让我感受到了友谊的珍贵。感谢我的初中高中老师们，是你们在我成长的道路上给予我最好的教育，让我在学业上不断进步，人格逐渐完善。

感谢一路走来遇到的每一位，是你们让我在成长的道路上不再孤单，让我学会尊重，学会谦逊，学会不耻下问，让我成为一个更好的人。

本课题承蒙国家自然科学基金（No. 62372264）资助，特此致谢。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：任家纬　　日　期：2019年5月27日

综合论文训练记录表

<table>
<tr><td>学生姓名</td><td>任家纬</td><td>学号</td><td>2020012377</td><td>班级</td><td>软件 02</td></tr>
<tr><td>论文题目</td><td colspan="5">自然语言到多类型查询语言的翻译方法研究</td></tr>
<tr><td>主要内容以及进度安排</td><td colspan="5">主要内容：

1. 文献调研，确定选题；

2. 构造数据集并适配基线方法

3. 设计自己的 SESD 方法

4. 设计演示系统。

进度安排：

1. 1~3 周，构造数据集；

2. 4~6 周，适配基线方法并完成对应实验；

3. 7~9 周，设计本文方法并完成对应实验；

4. 10~13 周，设计演示系统并完成论文。

指导教师签字：_____

考核组组长签字：_____

2024 年 1 月 17 日</td></tr>
<tr><td>中期考核意见</td><td colspan="5">考核组组长签字：_____

2024 年 4 月 7 日</td></tr>
</table>

指导教师评语	任家纬同学在综合论文训练中，深入研究了从自然语言到多类型查询语言的翻译方法，具体包括提出了自然语言到多类型查询语言（NL2PQL）的任务，适配了 NL2PQL 任务的基准数据集和基线方法，以及提出了解决 NL2PQL 任务的面向多类型查询语句的类型和结构信息的翻译方法。任家纬同学已经具备运用所学专业知识探索和解决问题的能力，同意其获得清华大学学士学位。 指导教师签字：_____ 2024 年 6 月 6 日
评阅教师评语	在综合论文训练过程中，任家纬同学针对自然语言向多类型查询语言转换的课题进行了深入研究，成功构建了适用于该任务的基准数据集和基线模型，并设计了一种专注于多类型查询语句的类型和结构信息的创新翻译方法，最后通过实验验证了上述模型与方法的有效性。论文工作量符合本科毕业设计要求。 评阅教师签字：_____ 2024 年 6 月 6 日
答辩小组评语	答辩小组组长签字：_____ 2024 年 6 月 7 日

总成绩：_____

教学负责人签字：_____

年　　月　　日