分类号_____ 学号 M202077062

学校代码 10487 密级_____

华中科技大学

硕士学位论文

（学术型□ 专业型 ☑）

面向中文自然语言转结构化查询语句关键技术研究

学位申请人：彭付强

学 科 专 业：计算机技术

指 导 教 师：魏巍 副教授

答 辩 日 期：2022 年 5 月 30 日

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Master Degree in Engineering

Research on Key Technologies for Chinese Natural Language to Structured Query Language

Candidate :PENG Fuqiang
Major :Computer Technology
Supervisor :Assoc.Prof. WEI Wei

Huazhong University of Science and Technology
Wuhan 430074, P. R. China
May, 2022

摘 要

关系型数据库在大数据时代承载了大量信息，其交互方式主要依赖于结构化查询语言（Structured Query Language，SQL），但对于非专业用户来说，难以利用 SQL 语言准确表达自己的查询需求。自然语言转 SQL 语句（Natural Language to SQL，NL2SQL），能准确地将自然语言描述的查询需求自动化地转换成 SQL 语句，以有效提升关系型数据库查询的便捷性，其具有重要的研究意义和商业化应用价值。

目前，已有 NL2SQL 技术仅针对英文文本数据，而中文 NL2SQL 研究相对滞后，其中主要面临的挑战之一是如何准确识别中文文本数据中数据库表格的列名信息，以及自然语言描述中口语化表述与数据库中数据表述不一致等问题。因此，考虑利用基于深度学习的自然语言处理技术解决中文 NL2SQL 任务中上述问题，主要工作如下：结合深度学习分类模型和基于转换器的双向编码表征（Bidirectional Encoder Representation from Transformers，BERT）预训练语言模型，以提升中文 NL2SQL 任务中 SQL 语句的生成准确性，其主要包括两部分：首先建立分类模型用于生成 SQL 语句中除条件值以外的内容，该部分考虑引入自然语言问句和数据库中表头等先验信息，并使用了适配中文 NL2SQL 模型的子任务，以有效解决列名缺失，多个查询条件关系连接错误，属性值和列值未能对齐等问题；然后建立分类模型用于生成 SQL 语句中条件值，该部分是在第一部分识别结果上约束列值类型并引入数据库表内容，采用分类标签的方法生成条件值，以有效解决自然语言描述与数据库数据表述不一致的问题。

通过在大规模真实数据集上实验显示，所提模型能够达到 87.92%的准确率，证明文中方法能够有效解决中文 NL2SQL 任务中数据库元数据识别不准确和自然语言口语化表述导致的不一致等问题，从而有效提升了中文 NL2SQL 算法生成 SQL 语句的准确性。

关键词： 中文 NL2SQL；自然语言处理；预训练语言模型；分类模型

Abstract

Relational databases carry a large amount of information in the era of big data, and their interaction mainly relies on Structured Query Language (SQL), but it is difficult for non-expert users to accurately express their query requirements using SQL language. Natural Language to SQL (NL2SQL), which can accurately and automatically convert the query requirements described in natural language into SQL statements to effectively improve the convenience of relational database queries, has important research significance and commercial application value.

At present, the existing NL2SQL technology is only for English text data, while the research on Chinese NL2SQL is relatively lagging behind. One of the main challenges is how to accurately identify the column name information of database tables in Chinese text data, and the inconsistency between the spoken expressions in natural language descriptions and the data expressions in the database. Therefore, deep learning-based natural language processing techniques are considered to solve the above problems in Chinese NL2SQL tasks. The main work is as follows: combining deep learning classification models and Bidirectional Encoder Representation from Transformers (BERT) based pre-trained language models to improve the accuracy of SQL statement generation in Chinese NL2SQL tasks, which consists of two main parts. Firstly, a classification model is established for generating contents other than conditional values in SQL statements, and this part considers the introduction of a priori information such as natural language questions and table headers in the database, and uses subtasks adapted to the Chinese NL2SQL model to effectively solve problems such as missing column names, wrong connection of multiple query condition relations, and failure to align attribute values and column values. Secondly, a classification model is established for generating conditional values in SQL statements. The conditional values, which are constrained column value types on the first part of recognition results and introduced into the database table content, are generated using the classification tagging method to effectively solve the problem of inconsistency between natural language description and database data representation.

Experiments on a large-scale real dataset show that the proposed model can achieve 87.92% accuracy, proving that the method in the paper can effectively solve the problems

of inaccurate database metadata identification and inconsistency caused by natural language colloquial expressions in Chinese NL2SQL tasks, thus effectively improving the accuracy of SQL statements generated by the Chinese NL2SQL algorithm.

Keywords: Chinese NL2SQL, Natural Language Processing, Classification Model, Pre-trained Model

目　录

1 绪论

1.1 研究背景与意义

随着互联网信息技术的快速发展，人们日常生活中产生了海量数据，其中大部分都存放在结构化或半结构化关系型数据库中，以便于集中管理和利用。目前，数据库中存储数据的统计分析和应用主要通过结构化查询语言[1]（Structured Query Language，SQL）等编程语言来实现，其要求使用者必须具备相应的基础数据库使用知识，而对非专业用户来说，即使是数据库数据的简单操作也是相当困难的。因此，如何通过自然语言的交互方式实现与数据库的人机交互成为了目前亟需解决的重要研究问题之一，其具有广泛的商业化应用价值和重要的研究意义。

在自然语言处理[2]（Natural Language Processing，NLP）研究领域，已有广大研究学者开始关注自然语言转 SQL 语句[3]（Natural Language to SQL，NL2SQL）方面研究，其可以有效将人类语言[4]转化为计算机可运行的数据库查询语言。单从技术角度来看，NL2SQL 能将用户的自然语言自动化转换为计算机可理解、可运行的数据库语言，其是自然语言处理领域语义解析[5][6]（Semantic Parsing）中重要子任务之一，其能够有效在非结构化自然语言和结构化数据库语言之间构建桥梁。目前，随着人工智能技术的蓬勃发展，基于自然语言方式的人机交互需求快速增长，同时自然语言处理领域前沿研究和技术在不断变革，上述因素均为 NL2SQL 的研究提供了良好的前提条件。

目前，关系型数据库具有广泛的应用场景，其中存储了各行各业海量数据，而其中的数据库结构通常十分繁杂，因此 NL2SQL 技术能够有效帮助人们简单便捷的对数据库进行操作，大大降低了数据库系统使用门槛，即便是非专业用户也可以简单快速地完成相应的数据检索与数据分析需求。因此，对 NL2SQL 技术的深入研究不仅能帮助企业节约高昂的数据库培训费用，也能使普通用户在不需要了解数据库系统中复杂表结构等信息前提下，高效准确的完成各种复杂的数据库操作需求，从而有效提高用户查询准确率和工作效率。

近年来，随着深度学习技术快速发展，已有研究将其应用于自然语言处理领域不同任务中均取得了不错性能[7]，因此基于深度学习的 NL2SQL 技术也日益受到广大研究学者的关注。但总体来说，目前的 NL2SQL 技术大多仍处在研究初期，且大部分研究仅针对英文数据库系统，而中英文语系在自然语言处理技术上存在天然差异，因此面向英文的 NL2SQL 模型难以简单直接应用于中文 NL2SQL 任务中。因此，面向中文 NL2SQL 的研究具有重要研究意义和挑战，对其深入研究有助于促进基于自然语言的人机交互领域技术的快速发展，同时也可为其他语义分析领域的相关技术发展提供可借鉴意义。因此，本课题将基于现有英文 NL2SQL 模型和方法，在已有中文 NL2SQL 数据集上开展研究，重点研究如何将自然语言自动转化为可执行的 SQL 语句，从而有效提升面向中文的 NL2SQL 技术的正确率，其对后续算法的研究和相关系统开发具有促进意义。

1.2 国内外研究现状

目前，已有 NL2SQL 相关研究可大致分为两类，分别是在技术相对受限的情况下基于规则的方法和基于深度学习模型的 NL2SQL 技术。与上述两种方法相关的研究工作具体介绍如下。

1.2.1 基于规则的方法

NL2SQL 最早可以追溯到上世纪中后期，研究学者已开始关注如何利用自然语言操作数据库的解决办法。比如早期的 NLIDB[8][9]（Natural Language Interfaces to Databases）方法，其可通过自然语言简单读取数据库系统中存储的内容。到了 70 年代初，著名的 LUNAR[10]系统被提出，但该系统主要用于解决特定地质学数据分析问题。同期，Winogand[11]的 SHRDLU 系统和 Schank[12]的 MARGIE 系统被提出，但上述系统不具有通用性，仅针对特定数据库系统，因而具有较大局限性。此后，有研究学者提出 Ladder[13]框架，其通过修改语义和语法信息以适应不同领域数据库查询需求，在一定程度上可解决不同领域数据库迁移问题，但仍需在相关领域重新定义新的解析语义语法技术用于对数据库进行操作。上世纪 80 年代中期，主要使用自然语言与数据库交互的方式解决接口移植问题，其中代表性的系统包括如 TEAM[14]等，其

主要通过系统界面与数据库管理人员交互式对话来获得数据库信息，并根据数据库信息访问特定数据库中存储数据。但当时受限于诸多因素影响，如系统语言支持上限、语义和语言逻辑歧义、语言理解上限、生僻词等，NL2SQL 技术发展缓慢。

到 21 世纪，基于规则的 NL2SQL 技术性能得到大幅提升，如基于自然语言接口（Natural Language Interfaces，NLI）技术等。后续，NLI 系统被大致可分为四类[15]：基于关键字的系统（Keywords-based Systems）、基于模式的系统（Pattern-based Systems）、基于句法分析的系统（Parsing-based Systems）以及基于语法的系统（Grammar-based Systems）。

（1）基于关键字的系统。此类系统主要利用关键字完成数据查询，在数据查找过程中，系统尝试将自然语言问句中字词与数据库中存储内容的倒排索引进行比较分析。为降低自然语言问句中部分噪音（如字词等）干扰，此类系统通过将问句中停用词删除或者采用输入关键字的方式，代表性系统包括 SODA[16]、SINA[17]和 Aqqu[18]等。

（2）基于模式的系统。该类系统以基于关键字的系统为基础，能够有效处理更宽泛的基于自然语言的查询需求，已具备回答复杂问题的基础能力。此类系统不再限制使用者输入的问句在给定的查询范围之内，因此具有一定通用性，代表性系统包括 NLQ/A[19]、QuestIO[20]等。

（3）基于句法分析的系统。基于句法分析的系统比基于关键字的/基于模式的系统表现更为出色，该类系统利用语义解析器对输入的自然语言查询问句进行分析处理，从而更好的理解输入问句的语义信息，并能够将其映射为 SQL 语句，代表性系统包括 ATHENA[21]、BELA[22]和 BioSmart[23]等，但其在 SQL 语句转化准确性以及自然语句理解能力方便仍稍显不足。

（4）基于语法的系统。基于语法的系统的核心利用预定义的语法规则限定系统能够理解和回答的问题范围。根据定义规则，通过使用符合语法规则的自然语言问句，让系统能够准确理解和识别用户输入的查询需求，从而有效提升了系统交互的准确性和效率，代表性系统包括如 AskNow[24]、SPARKLIS[25]和 GFMed[26]等，但预定义规则不具备完备性，同时仍需用户提前了解相关语法规则，且手工编写规则大大降

低了人机交互的效率。

1.2.2 基于深度学习的方法

以往基于传统方法的工作虽然能够有效处理部分简单查询需求，但面对复杂查询需求时，传统方法需要应用预定义语法规则解析相关自然语言表达的查询需求以生成对应的 SQL 语句，因此不具有通用性，即无法满足 NL2SQL 任务对领域适应性的要求。随着深度学习的兴起，研究学者开始将深度学习技术应用到 NL2SQL 任务中，促进了 NL2SQL 技术进一步发展。

传统序列到序列模型[27][28]（Seq2Seq）的解码器部分使用固定单词表，即从已定的单词表中检索出生成序列，但自然语言转 SQL 语句不是简单的序列生成任务，因此难以直接利用 Seq2Seq 模型完成 NL2SQL 任务。Vinyals 等人[29]在 2015 年提出的 Pointer Networks 有效解决了这一问题，该网络使用注意力机制在输入序列中选择单词作为输出，其输出词表是随输入序列的改变而变化的，NL2SQL 任务利用此特点将自然语言问句及目标 SQL 语句中可能出现的其他词作为输入序列，从而解决了 Seq2Seq 模型存在的问题。Dong 等人[30]在 2016 年以编码器-解码器为基础提出一种模型，该模型将自然语言编码作为向量表示，并提出 Seq2Seq 和 Seq2Tree[31]两种模型变体，其中 Seq2Seq 模型能将 NL2SQL 任务简化为序列生成任务，而 Seq2Tree 模型则能将解码器修改为层次树的结构。Zhong 等人[32]在 2017 年基于深度学习方法提出 Seq2SQL 模型来解决自然语言转 SQL 语句问题。Seq2SQL 模型将生成的 SQL 语句分为三个部分：AGG、SELECT、WHERE，每个部分分别使用不同的方法进行计算，其中 SELECT 部分与 AGG 部分均采用注意力机制进行分类，WHERE 部分则利用 Pointer Networks 进行训练，同时该模型使用强化学习[33]优化生成结果。除了 Seq2SQL 模型，Zhong 和其他研究者发布了首个人工标注数据集 WikiSQL[34]，该数据集同时也是首个大规模英文 NL2SQL 数据集。基于 Seq2SQL 模型，Xu 等人[35]在 2017 年提出 SQLNet 来解决 Seq2SQL 模型中强化学习效果不明显的问题，该模型的基本思想是插槽填充，即利用 SQL 语句结构特征将其分割为不同部分，将各部分预测结果分别填入槽内得到完整 SQL 语句。Yu 等人[36]提出的 TypeSQL 模型以 SQLNet 模型为基础在建模方法上进行改进，SQLNet 模型对 SQL 语句分割后的每个部分都

设定单独的模型，TypeSQL 则摒弃了单独建模的方式，而是对相似部分使用统一的神经网络模型，从而简化了模型结构。Dong 等人[37]在 2018 年提出一种结构感知神经架构，该架构将 SQL 语句生成分为两阶段，先根据自然语言问句生成粗略结果，然后通过自然语言问句和粗略结果本身填充其缺失部分，此方法能有效提高模型性能。在预训练模型[38][39]被使用之后，一些研究学者开始采用预训练模型方式来进行语义解析，这也为 NL2SQL 技术发展提供了基础。在 2019 年，Wonseok 等人[40]基于预训练模型及 WikiSQL 数据集设计了 SQLova 模型，该模型执行准确率在 WikiSQL 数据集已接近上限。但 WikiSQL 数据集数据表结构较简单，所涉及到的自然语言问句对查询要求较低，因此其大多数 SQL 语句无法应用于实际场景。

为解决 NL2SQL 任务对大型高质量数据集的需求，Yu 等人[41]在 2018 年推出第二个大规模英文 NL2SQL 数据集 Spider，该数据集包含跨领域 SQL 语句，与 WikiSQL 相比更贴近真实场景。在提出 Spider 之后，Yu 等人[42]提出 SyntaxSQLNet 模型，该模型基于 Spider 数据集并采用树形语法结构，使用递归的形式来管理 SQL 语句的生成，并在论文中给出使用数据增强方法来提升模型表现的具体流程。在 2019 年，Lee 等人[43]发表的论文中提出了新模型 RCSQL，该模型主要做了以下几点：（1）根据不同 SQL 语句创建解码器；（2）使用序列到集合（Seq2Set）方法取代 Seq2Seq 方法对 SQL 语句中的列值进行预测；（3）使用循环递归方法生成子查询语句。Bogin 等人[44]为能够高效利用关系型数据库结构信息，将数据表格的各个列视作节点，表格内各节点之间存在联系，根据数据库表的此种特性，该模型使用门控图神经网络[45][46]（GGNN）来解决 Spider 中的多表问题。不同于先前端到端[47]（End-to-End）合成 SQL 语句的方法，Guo 等人[48]发表的 IRNet 将 SQL 语句合成过程分为三阶段：在第一阶段，该模型对当前自然语言问句与数据库结构信息之间进行模式链接；在第二阶段该模型使用基于语法的神经网络模型去合成 SemQL[49]，SemQL 是介于自然语言问句与 SQL 语句之间的中间语言；在第三阶段，该模型根据领域知识从上一阶段合成的 SemQL 查询中推断出可执行 SQL 语句。为继续提高 NL2SQL 模型表现，Wang 等人[50]提出了 RATSQL，该模型可视作图神经网络的后续工作，论文中首先对数据库模式编码和数据库模式链接进行介绍，其次使用关系感知自注意力机制[51][52]（Relation-

Aware Self-Attention）来处理，将自然语言语句与数据库信息之间的显式关系和隐式关系同时考虑进编码器中，以此完善 RATSQL 模型的表示能力。

随着国内对深度学习研究的兴起，很多研究学者尝试使用深度学习的方法进行自然语言向 SQL 语句的转换。刘译璟[53]将传统的语法分析方法和深度学习模型相结合来构造 NL2SQL 模型，首先用深度学习模型得到解析结果，然后在依存句法树中将深度学习模型的结果标记在依存语法树的节点上作为属性，对于常规问题，语义解析方法能够对深度学习模型生成的结果进行补充和纠正，可以达到比较好的效果。张琰[54]采用了当前深度学习领域性能较好的基于转换器的双向编码表征[55]（Bidirectional Encoder Representation from Transformers，BERT）模型进行数据增强表示，针对 19 年发布的首个中文自然语言转 SQL 数据集，构造了深度学习模型，取得了较好的效果。

上述工作将 NL2SQL 模型在 WikiSQL 数据集的准确率从最初的 59.4%提升到 91.8%，针对 Spider 数据集构建的模型准确率也从最初的 12.4%提升到 61.9%，所采用的方法也从传统方法到简单的 Seq2Seq、深度学习、预训练等。使用基于深度学习的方法解决 NL2SQL 任务时，关键点即如何正确识别自然语言所包含的数据库表内容信息和列名信息。

1.3 本文主要研究内容

通过调查 NL2SQL 国内外相关研究，发现目前该领域的研究工作主要集中在英文，中文 NL2SQL 研究工作较少，由于中文语义复杂性以及中文 NL2SQL 数据集的缺乏，使得中文 NL2SQL 研究工作相比于英文 NL2SQL 更为困难。本文基于已有 NL2SQL 工作将中文 NL2SQL 任务分为多个子任务，根据子任务特点分别进行建模和训练，最后利用 SQL 语句语法规则对所有子任务预测结果进行拼接，得到完整可执行的正确 SQL 语句。本文主要研究内容如下所述。

（1）基于已有 NL2SQL 技术，将深度学习文本分类模型和 BERT 预训练语言模型结合，对自然语言问句和数据库表格列名信息进行语义分析。

（2）根据 SQL 语句语法规则将 SQL 语句的生成过程分为两步：首先建立分类

模型抽取 SQL 语句中出现的列名、聚合操作、运算符、条件语句以及条件语句之间的连接符等，为解决 NL2SQL 任务中列名重复使用这一难点，模型中单独设置子任务预测列名是否重复使用；其次建立分类模型抽取 SQL 语句中条件值，此模型结合前一模型的结果有效解决了自然语言描述与数据库数据表述不一致的问题。

（3）针对本文所提方法，选择具有代表性的模型在真实数据集上进行对比实验，评估本文所提方法和模型在中文 NL2SQL 任务中的有效性。

1.4 本文组织结构

全篇一共包含五个章节，每个章节的主要内容如下。

第一章是绪论，首先阐述 NL2SQL 任务的研究背景与研究意义，然后分析 NL2SQL 研究历史及研究现状，接着对本文主要研究内容进行介绍，最后给出全文组织结构。

第二章是中文自然语言转结构化查询语句问题分析，首先介绍 NL2SQL 技术应用及行业分析，然后详细分析已有 NL2SQL 相关工作和本文所使用的预训练模型，最后给出中文 NL2SQL 任务所涉及到的相关概念和方法。

第三章是中文自然语言转结构化查询语句方法，本章首先对中文 NL2SQL 任务进行介绍，然后详细说明本文如何利用自然语言问句信息以及数据表内容信息，最后介绍本文面向中文 NL2SQL 所提的改进方法及模型结构，并给出建模方法和损失函数设计方案。

第四章是实验结果与分析，本章首先对中文 NL2SQL 数据实例进行介绍，并说明本文对数据所做处理，然后选择已有 NL2SQL 模型与第三章所提模型进行对比实验，最后通过实验结果验证了本文所提模型的性能。

第五章是总结与展望，本章主要总结了本文所做的研究工作，并对 NL2SQL 技术未来发展方向进行展望。

2 中文自然语言转结构化查询语句问题分析

自然语言转结构化查询语句（NL2SQL）具有广泛的应用场景，目前该领域已有大量研究工作。本章将对中文 NL2SQL 相关问题进行分析，首先介绍 NL2SQL 的商业价值及其在实际应用时所存在的问题，其次对已有工作进行分析并提出针对中文 NL2SQL 问题可能的改进方向，最后介绍本文研究问题涉及的相关概念和方法。

2.1 相关应用分析

自然语言转 SQL 语句（NL2SQL）能将用户自然语言自动转化成计算机可运行的结构化查询语句，从而实现通过自然语言的方式与数据库进行人机交互。其目的是使用户在没有掌握专业数据库编程技术的情况下仍能够灵活快速地分析数据库所存储的数据，从而大大降低数据库系统应用门槛。

NL2SQL 在技术领域的本质是将自然语言问句转换成可理解、可操作、符合计算机运行规则的语义表示，需要使计算机理解自然语言，从而产生正确表述自然语言语义的可执行程序式语句。该技术是语义解析和智慧问答等领域的前沿问题，也是提升数据库数据分析效能的关键技术之一。

以往使用数据库中存储数据检索业务信息并做统计分析时，通常需要手动编写 SQL 语句。例如，对数据库表格进行查询时，针对该需求需要自行写出正确的 SQL 语句方能在数据库中执行并得到查询结果，如图 2.1 所示。

普通用户面对图 2.1 中查询需求时，往往需要求助专业人员或是学习相关专业知识才能进行查询操作，该方法提高了普通用户操作数据库的门槛，同时也降低了用户工作效率。NL2SQL 技术能自动将自然语言描述转换成 SQL 查询语句，然后再交由计算机去处理，从而节省人工构建 SQL 语句的所有步骤，也因此大大提高了人机交互效率。

结构化知识信息在日常生活中无处不在，因此 NL2SQL 技术具有相当广泛的应用范围，举例说明如下。

（1）保险行业企业内部经营数据的即时查询，如客户信息咨询服务；

（2）证券行业各种结构化数据的查询，如上市行研研报和财报；

（3）出行行业中旅游场景的智慧问答，如酒店旅游、机票、火车票行程查询；

（4）日常生活服务的智能问答，如电话费用查询、水电费用查询。

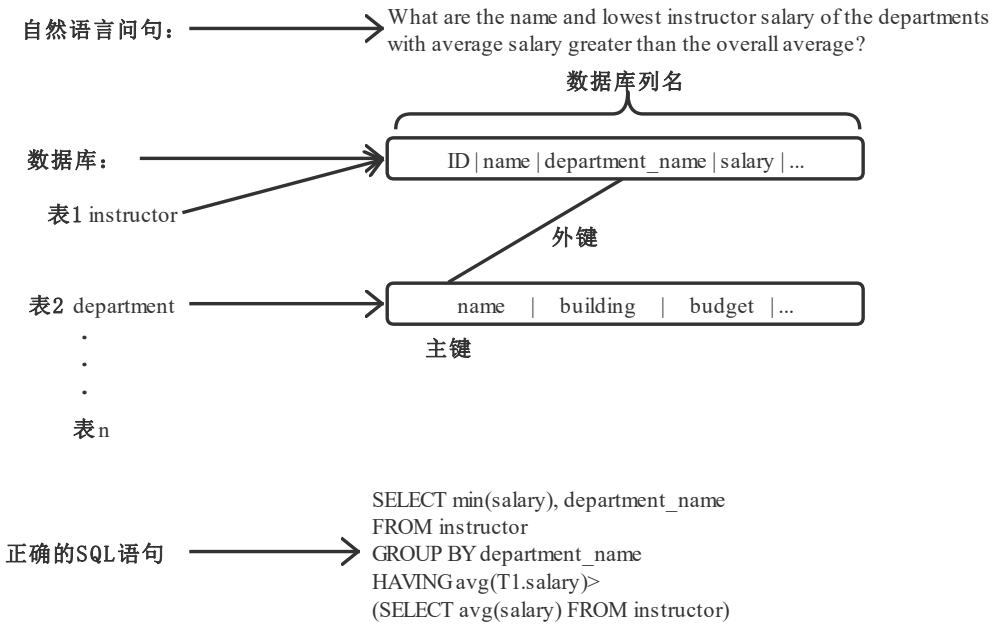

图 2.1　Spider 数据集自然语言问句及对应的 SQL

NL2SQL 的应用场景也十分广泛。例如以结构化信息技术为基础的人工智能互动和智能问答服务；对网络搜索引擎进行更深层次的优化；为数据库使用者提供便捷化智能服务。目前已有与 NL2SQL 相关的工业界落地应用产品出现，比如微软的 PowerBI、网易的七鱼智能客服机器人和阿里云智能数据助理等，都是基于 NL2SQL 技术而实现。

但目前 NL2SQL 的落地应用仍有很大的局限性，其研究工作与落地之间依然存在较大差距。本文通过对比分析研究界和实际应用所使用的数据发现，其差异主要有以下三点：首先是数据库表组织形式不同；其次是模型训练数据的差异；最后是数据自身的复杂性，比如数据库中数据列值的复杂关系使得列与列之间可能存在上下级关系。

因此，NL2SQL 的研究引起了工业界和学术界的普遍重视，而面向中文文本数据的 NL2SQL 研究基础相对薄弱，在当前阶段，中文 NL2SQL 的研究应获得更多研究

者的关注。

2.2 相关工作分析

在使用深度学习的方法解决 NL2SQL 任务之前，该任务通常使用传统的方法将自然语言转为 SQL 语句，比如通过高质量的语法树和词库构建语义分析器。其中语法树的样例图如图 2.2 所示。NLP 领域最广泛的两种句法分析理论分别是转换生成语法和依存句法，图 2.2 中的方法是转换生成语法，该方法根据句子文法产生式推导出句子短语结构树，也称为推导树。

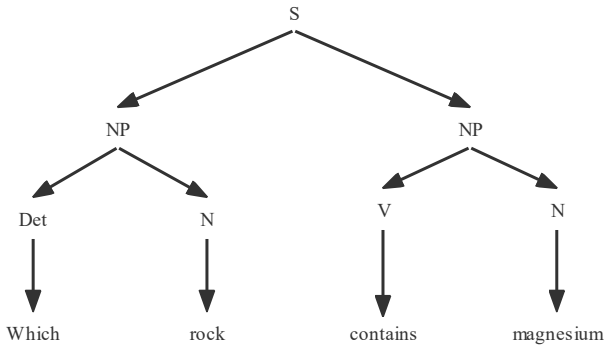

图 2.2 语法树示例图

在深度学习的方法被大规模使用之后，NL2SQL 任务主要的解决方案是将 End-to-End 模型与 SQL 语句功能规则进行结合。WikiSQL 数据集提供了超过 80 000 个标记的数据对，为 NL2SQL 模型提供了足够的训练数据。目前在 NL2SQL 任务具有良好性能的模型包括 SQLova 和 X-SQL，上述模型在 WikiSQL 测试集上各自获得 89.6% 和 91.8%的执行准确率。本文接下来对上述两种模型进行介绍，并分析有关中文 NL2SQL 的问题。

2.2.1 分析 SQLova 模型

SQLova[40]是通过进一步完善 SQLNet[35]的模型结构而衍生出来的 NL2SQL 模型，因此，首先分析 SQLNet 模型结构，其次分析如何在中文 NL2SQL 任务应用此模型。

SQLNet 模型是基于 WikiSQL 数据集的 NL2SQL 模型。该模型针对不同场景采用不同的序列到序列结构，以解决 Seq2Seq 模型在 SQL 生成过程中的问题。其论文中采用基于草图的方法，其中草图包含如图 2.3 中右边部分所示的依赖关系，可以只

考虑依赖的先前预测结果来进行下一个预测。首先根据 SQL 结构对 SQL 语句形式进行规范，如图 2.3 中左边部分所示。其中 SELECT、WHERE 以及 AND 表示 SQL 语句的关键字，$AGG 表示 SQL 语句的聚合操作类型，$COLUMN 表示数据表的列名，$VALUE 在此论文中表示输入自然语言问句中的一个子串，$OP 表示比较符号，结尾的“*”表示括号内的额片段可以出现 0 次及以上。SQLNet 对 WHERE 语句进行预测时将 WHERE 语句中列的数量作为约束，然后选取该数量的列。运算符和 SELECT 语句的预测都是分类问题。

SQLova 中借鉴了 SQLNet 中对 NL2SQL 任务进行分步解决的思想，但 SQLova 使用 BERT 作为模型的输入表达层，代替了词向量。除此之外，为了对数据库表中的列名和自然语言问句信息加以利用，SQLova 在编码的输入加入了列名和问句，提高了数据库表感知的单词上下文环境与 BERT 的结合程度。

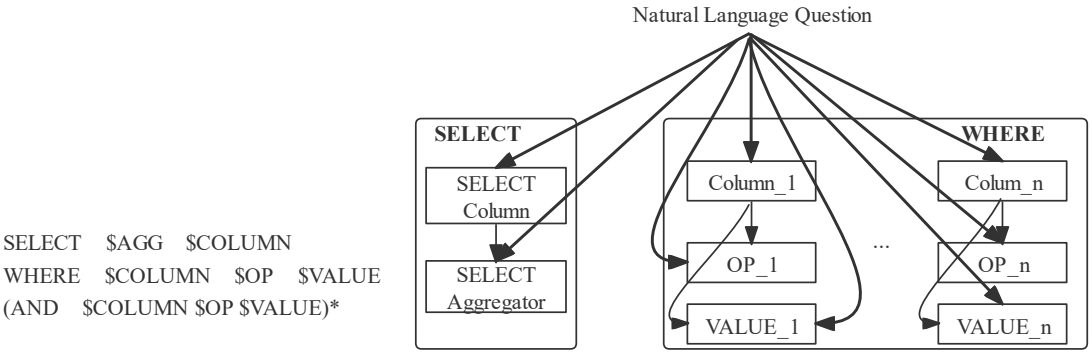

图 2.3 SQL 规范与依赖关系示意图

另外，SQLova 在子任务预测时基于 SQLNet 做了几个改进，其中包括条件值的预测。在预测条件值时，SQLNet 只把列名作为子模型的输入，而 SQLova 把列名和列操作的特征都作为子模型输入，提高了条件值预测的准确率。

2.2.2 分析 X-SQL 模型

X-SQL[56]模型使用了 BERT 风格的多任务深度神经网络[57]（Multi-Task Deep Neural Networks，MT-DNN）预训练模型的上下文输出来增强结构模式表示，并结合类型信息来学习用于下游任务的模式表示。该模型包含了三层结构，分别是 Sequence Encoder（序列编码器）、Context Enhanced Schema Encoder（语义增强模式编码器）、

Output Layer（输出层），三层架构的示意图如图 2.4 所示，其中最上面一层是输出层，输出层的作用是完成 SQL 语句各部分内容的预测任务，模型将其分为六个子任务，分别是 SELECT 语句列值选择（S-COL）、聚合操作（S-AGG）、WHERE 语句选择列数量（W-NUM）、WHERE 语句列值选择（W-COL）、WHERE 语句操作符（W-OP）和 WHERE 语句条件值（VAL），上述任务相互结合并且相互制约。

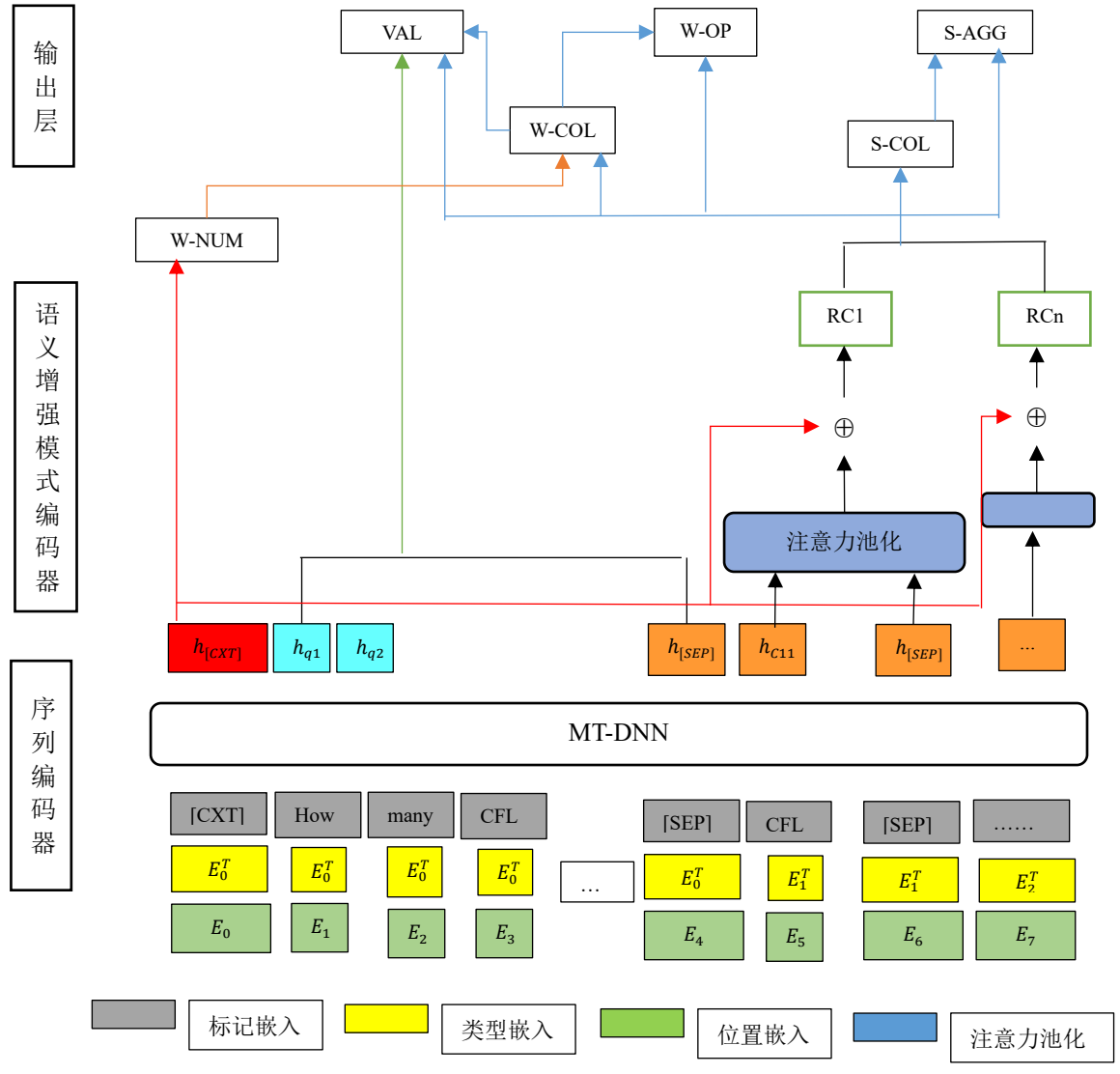

图 2.4　X-SQL 模型结构图

第一层是序列编码器，在该结构中，对每个数据表增加一个 Empty 列，然后段编码替换为类型编码，编码过程中对问句和数据表信息中的四种信息进行学习，分别是 Question（自然语言问句）、Categorial Column（范围列）、Numerical Column（数

字类）和 Empty Column（空列）。

第二层结构是语义增强模式编码器，该结构用于增强在 Sequence Encoder（序列编码器）得到的向量。

第三层是完成六个子任务预测任务的输出层，输出层的六个子任务中第一个子任务是 S-COL，表示 SQL 语句选择数据库表中的列名；第二个子任务 S-AGG 表示对第一个子任务的聚合操作；其余四个子任务 W-NUM、W-COL、W-OP 和 VAL 组成 SQL 语句中的 WHERE 语句。其中 W-NUM 表示对数据库表的几列进行约束，W-COL 表示对哪几列进行约束，W-OP 表示对上述列的操作符，VAL 表示被上述几列约束的条件值。其中 W-COL、W-OP 和 VAL 是依赖 W-OP 子任务的，VAL 的值是来源于自然语言问句的，并且各个子任务使用各自的损失函数分别优化，W-NUM、W-OP、VAL、S-COL 和 S-AGG 在训练期间均使用交叉熵作为模型的损失函数，而 W-COL 预测任务则采用基于 KL 散度的做法作为模型损失函数。

由于在输入和模型结构方面的优化，X-SQL 模型能够在 WikiSQL 数据集测试中达到 91.8%的最终准确率。因此，通过大型预训练模型作为输入编码，如 BERT 和 MT-DNN，仅靠原始解决方案就能在 WikiSQL 数据集上取得明显的改进。

2.2.3 模型实验测评

目前 NL2SQL 任务常见的数据集主要分为英文数据集和中文数据集两大类。其中英文数据集有 WikiSQL 和 Spider，中文数据集有 CSpider 和追一科技在 2019 年公开的用于自然语言转 SQL 语句数据集[1]，本文把后者称为中文 NL2SQL 数据集。

经过统计分析，WikiSQL 数据集已经可以满足研究学者设计 NL2SQL 模型时对数据量的需求，但 WikiSQL 数据集中 SQL 语句形式较为简单，并不涉及复杂查询语句，目前 SQL 执行结果准确率已有 91.8%，所以该数据集上的研究成果已经达到较高水平。中文 NL2SQL 数据集中约有 6 000 多张数据表，数据表共包含约 50 000 条自然语言问句和对应的 SQL 查询示例，与其他数据集相比，该数据集的数据量比较

[1] https://drive.google.com/file/d/1rGo7qhc8QBRMKAtxpYtNS2t2p7xuhUdv/view?usp=sharing

适中，并且该数据集的 SQL 语句查询是单表查询语句，没有涉及到多表查询的问题，所以 SQL 语句的结构复杂度也比较适中。

考虑到数据集难易程度，且本文重点研究内容是面向中文自然语言问句生成 SQL 语句，所以本文实验选择中文 NL2SQL 数据集作为训练数据。该数据集比 WikiSQL 数据集难度更高，涉及到更多样的查询方式，自然语言问句中可以包含多个查询的条件，也可以具有与数据库数据表述不同的表述方式。表 2.1 展示了上述两个 NL2SQL 数据集在各方面的对比。

表 2.1　数据集对比

数据集名称	中文 NL2SQL	WikiSQL
语言	中文	英文
SELECT 语句	多个列	一个列
WHERE 语句	多个为主	单个为主
列名是否会被重复使用	是	否
条件连接符	[AND, OR]	AND
训练集	41 522	56 355
验证集	4 396	8 421
测试集	4 086	15 878

尽管上述分析的 NL2SQL 工作取得了较好的成果，但 SQLova 模型和 X-SQL 模型是面向英文的，而本文工作是解决面向中文 NL2SQL 问题。所以为了测验使用英文 NL2SQL 数据集训练得到的模型是否可以适用于中文 NL2SQL 问题，本文对中文 NL2SQL 数据集中的测试集和验证集数据进行标注之后，使用 WikiSQL 数据集和中文 NL2SQL 数据集对 SQLova 模型和 X-SQL 模型进行测验，结果如表 2.2 所示。

表 2.2　对比数据集测验结果

模型名称	数据集名称	DevAccLA	DevAccEA	TestAccLA	TestAccEA
SQLova	WikiSQL	81.6%	87.2%	80.7%	86.2%
X-SQL	WikiSQL	86.2%	92.3%	86.0%	91.8%
SQLova	中文 NL2SQL	80.3%	83.2%	81.3%	82.6%
X-SQL	中文 NL2SQL	82.5%	86.7%	81.8%	86.4%

本次测验的评估指标主要是逻辑准确率和执行准确率，逻辑准确率是指 NL2SQL

模型生成的 SQL 语句在数据集中真实 SQL 语句中的匹配程度，而执行准确率表示的是 NL2SQL 模型生成 SQL 语句的执行结果在真实 SQL 语句执行结果中的比例。在表 2.2 中，DevAccLA 表示验证集上的逻辑准确率，DevAccEA 则表示验证集上的执行准确率，TestAccLA 表示测试集上的逻辑准确率，TestAccEA 则表示测试集上的执行准确率。表 2.2 中测验结果与 SQLova 模型和 X-SQL 模型论文中的结果相比，SQLova 模型和 X-SQL 模型在 WikiSQL 数据集上的准确率均比本文在中文 NL2SQL 数据集上的准确率高，所以面向英文的 NL2SQL 模型直接在中文 NL2SQL 任务中使用时，模型效果会受到一定影响。

2.2.4 分析预训练语言模型

BERT 是谷歌公布的具有强大特征提取能力的预训练语言模型，其在多项 NLP 任务中均具有最佳表现，基于 BERT 的深度学习模型也成为 NLP 领域最具突破性的模型之一。本文为提高中文 NL2SQL 模型性能，考虑引入 BERT 模型来解决中文 NL2SQL 任务中存在的中文分词问题和数据库链接问题，下面详细介绍此模型的使用原理。

在 BERT 之前，预训练语言模型均是单向特征表示的自回归模型，单向模型限制了其预训练效果。而 BERT 能够执行深层的双向编码任务，训练时可以学习到双向语言表达，这也是 BERT 如此有效的主要原因。BERT 使用了新的训练方法解决双向编码干扰模型性能的问题，该方法即使用 Transformer[58]编码器学得双向表达以实现双向编码的掩码语言模型[59]（Masked Language Model，MLM）。MLM 与一般语言模型不同之处在于其将自回归方法改进为自编码方法，即通过随机 Mask 句子中某些字，将其视为预测任务并通过上下文进行预测。

在 BERT 的预训练过程中，模型会重复利用同一句子作为输入来更新模型参数，每个句子会采用 MLM 中随机 Mask 的方式，选择其中 15%的单词作为 Mask 内容。当重复输入此句子时，模型会采取以下三种处理方式：第一种是以 80%的比例将 Mask 内容用[Mask]替换，以此来表示需要预测的内容；第二种是以 10%的比例将 Mask 内容用随机词替换；第三种是余下 10%的比例使用原单词。示例句子如下。

（1）80%：I have a red ball → I have a [Mask] ball

（2）10%：I have a red ball → I have a yellow ball

（3）10%：I have a red ball → I have a red ball

BERT 的编码器使用了基于自注意力机制的 Transformer 编码器，其自注意力机制更容易捕获句子中长距离相互依赖的特征，利用此特征能够对所有单词进行深度编码且不受其位置影响，从而达到有效提取上下文信息的目的。

BERT 模型为输入序列设计了一组特定的规则，如图 2.5 所示。其中，输入序列开头设为特殊标记[CLS]，[SEP]标记则置于句子序列尾部用来表示该句子结束，同时也用来分割不同的句子序列，在后续模型训练时也可根据此标记区分不同文本向量。在多个编码层叠加的情况下，输入序列的[CLS]标记会经过深层次编码，在最高编码层输出整个句子对表示。基于 Transformer 的 BERT 模型会在编码过程中整合序列中所有信息，输出层的[CLS]标记对应的输出向量会作为所有输入序列的语义表示，并用于文本分类，因此 BERT 在梯度反向传播过程中会对分类相关的特征起到关键作用。

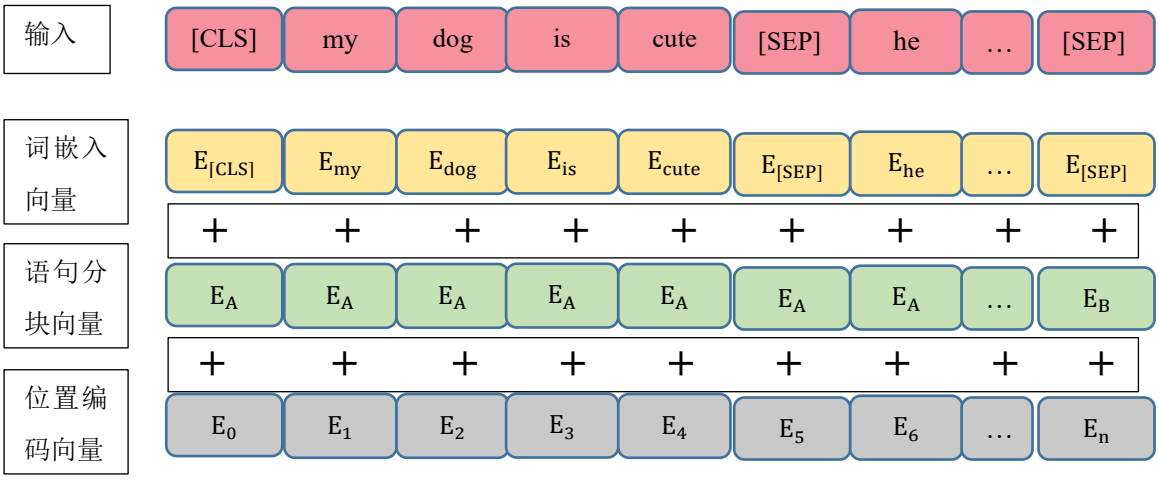

图 2.5　BERT 模型输入示意图

最终 BERT 的语言输入表示包含三个组成部分：词嵌入向量（Word Embeddings）、语句分块向量（Segmentation Embeddings）和位置编码向量（Position Embeddings），最终的 Embeddings 是三个向量之和。位置编码可以弥补 BERT 模型所使用的 Transformer 无法利用文本顺序信息，用数学方式来更清楚地表达：设模型为$y = f(x)$，其中输入词序列$x = [x_1, x_2, \dots, x_n]$，输出为$y$，对$x$进行随意置换$x' = [z_1, z_2, \dots, z_n]$依

然有$f(x) = f(x')$。因此，必须引入额外的位置编码向量来对文本顺序进行建模。

目前将预训练语言模型应用于下游任务主要有两种方式，第一种是将预训练语言模型所得的词向量作为下游任务的输入特征；第二种是在已训练好的语言模型基础上更新与下游任务相关的参数。上文介绍的 BERT 模型是使用第二种方式完成下游任务的预训练模型，本文基于 BERT 模型的特性将 NL2SQL 任务的上游任务设为分类任务，并基于 BERT 模型的迁移接口决定对 NL2SQL 任务建模的策略。

基于 X-SQL 模型的思路，本文对中文 NL2SQL 任务的建模有了基本架构，如图 2.6 所示，建模时将子任务视为分类任务，每个输入的句子经过 BERT 模型后，将模型最高层输出所得到的向量表征分别通过对应的全连接层即可得到相应的结果。

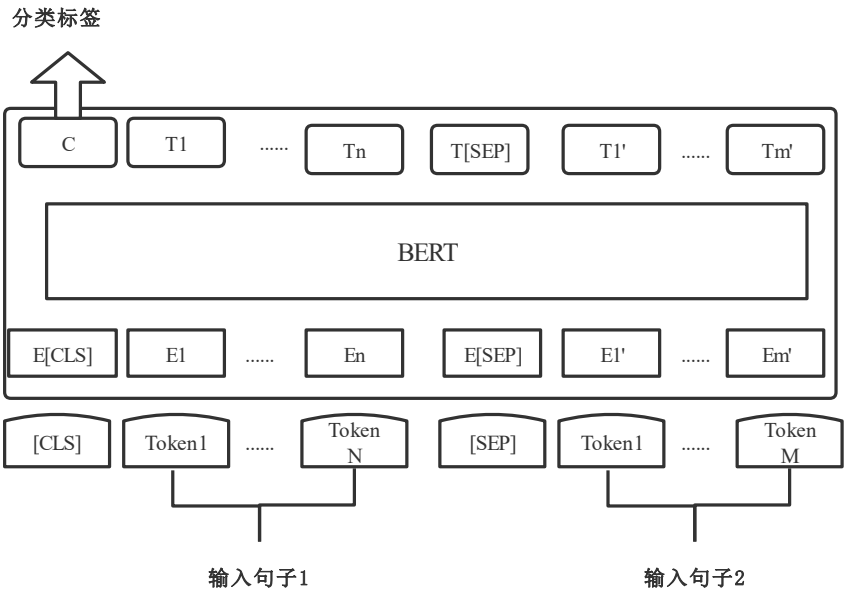

图 2.6　基于 BERT 的分类模型架构图

原始 BERT 模型是基于英文语境的，其分割字符的方式是将单词分割为若干英文字符，此方式下的样本生成方式是将分割后的字符随机 Mask。而 Chinese-BERT-wwm[60]（Whole Word Masking）权重，是基于原始 BERT 针对中文特性改进了句子 Mask 方式，并根据中文和英语语境的差异，将 Mask 方式改为完整词汇 Mask，其 Mask 策略与上文中掩码语言模型中对 Mask 部分的处理方式相同。生成全词 Mask 的例子如图 2.7 所示，这种中文自然语言处理中所使用的以词为粒度的中文分词方式能更好地学习中文字符向量表示。

因此在本文中，放弃了原始 BERT 模型对自然语言查询语句进行编码的方法，而是使用在 Chinese-BERT-wwm 预训练模型上进行微调的方法。原因之一是由于原始 BERT 模型是逐字分割，没有使用整个词的信息，所以无法学习到中文字词的连贯性和语义信息，而 Chinese-BERT-wwm 模型可以弥补此缺陷。此外，Chinese-BERT-wwm 预训练模型在语言表征和特征提取方面更有优势。

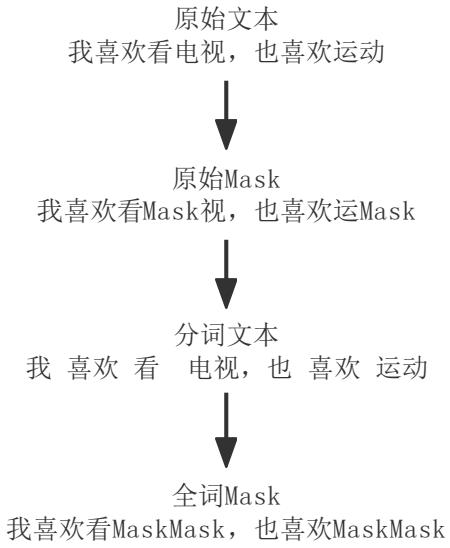

图 2.7　全词 Mask 生成样例

2.3　研究问题与方法

本文主要讨论中文 NL2SQL 问题，但是很少有研究清楚地说明 NL2SQL 问题的相关定义。因此，为更清楚描述本文如何开展中文 NL2SQL 研究，下面将提供中文 NL2SQL 相关符号的定义，同时也介绍本文根据现有理论知识提出的方法与流程。

2.3.1　相关符号定义

定义 2.1：中文自然语言问句：由祈使句或问句组成的中文文本，有明确的询问目的。在本文中，中文的自然语言问句是按照公式(2.1)来定义的。

$$Q = \{c_1, c_2, \dots, c_i, \dots, c_n\} \tag{2.1}$$

其中，Q是表示中文自然语言问句，n表示自然语言问句中的总字符数，c_i表示第i个字符，中文中字符可以是数字，英文字母，标点符号或者是汉字。表 2.3 中的句子是

查询数据库时的问句实例。

表 2.3　中文自然语言查询实例

Question1	你好，你帮我查一下 18 级旅管 1 班和 2 班一共有多少人
Question2	你好，请告诉我广州中山大学附属口腔医院的地址谢谢
Question3	如果是本科及以上学历的话，那可以报考哪些职位

定义 2.2：数据库表：数据库表中每条数据都有相同的属性值，数据通常属于同一类型，数据库表的定义在如公式(2.2)所示。

$$T = \{t, (h_1, h_2, \dots, h_i, \dots, h_m)\} \tag{2.2}$$

其中T表示数据库中存放的数据表，t表示该数据表T的名称，h_i表示表T中第i列的名称，m表示列的总数。表 2.4 给出了数据库表格的例子，其中第一行表示当前表的所有列名，其他部分是该数据表中的数据。

表 2.4　数据表实例

模拟组合	持股数量	最新股价	最新市值	最新市值比重
万科	365.0	8.85	32.0	31.2
金地集团	537.0	6.71	35.0	34.1

定义 2.3：数据库表集合：特定数据库中所有数据库表格的集合。在本文中，根据公式(2.3)对数据库的表集合进行形式化定义。

$$S = \{T_1, T_2, \dots, T_i, \dots T_k\} \tag{2.3}$$

其中，S表示数据库表格集合，T_i表示集合中第i个数据库表格，k表示数据库的表集合中表的总数。

定义 2.4：NL2SQL 查询任务：从给定的自然语言问句Q以及给定的数据表格T中生成一条完整的 SQL 语句，其定义如公式(2.4)所示。

$$(Q, T) \rightarrow R_{single} \tag{2.4}$$

其中自然语言问句Q与对应的数据库中表格的集合T共同构成公式(2.4)的输入，与输入中的问句对应的完整 SQL 语句R_{single}被定义为输出，其格式模型如图 2.8 所示。

图 2.8 中“#”后面的内容都是要填写的内容，COLUMN 表示 SQL 语句查询数据表的列名，自然语言问句中数据表的列名都可能被选中；AGG 代表对 COLUMN 使用哪种聚合函数操作，比如 MIN、MAX；而 OP 则代表 WHERE 条件后面选中的

列和列值的关系符合，比如“>”、“<”、“=”等；COND_OP 表示有多个 WHERE 条件时，条件之间的关系连接符，比如“AND”、“OR”和”NONE”；“*”表示括号内的内容出现多次，在“[]”外面表示方括号里面的内容至少出现一次，“()”外面表示括号里面的内容至少出现 0 次。

```
SELECT      [#AGG      #COLUMN]*
FROM        T.t
WHERE       #COLUMN   #OP   #VALUE
($COND_OP   #COLUMN   #OP   #VALUE)*
```

图 2.8　SQL 查询语句格式

2.3.2　方法与流程

图 2.9 处理中文 NL2SQL 任务的研究框架和主要流程，其中图中左半部分是数据处理方法与流程，右半部分是通过 NL2SQL 模型生成 SQL 语句的方法与流程。下面说明与图 2.9 有关的输入数据。

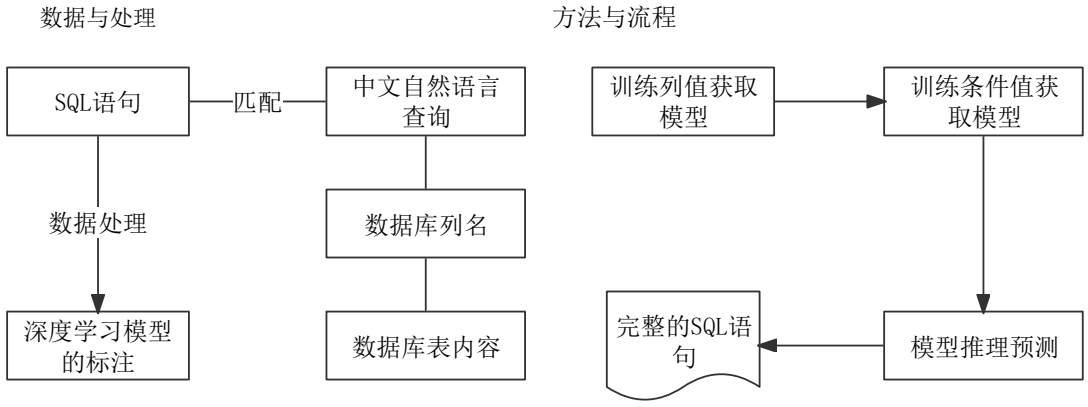

图 2.9　NL2SQL 处理方法与流程

（1）中文自然语言查询：对应于定义 2.1 中描述的中文自然语言问句；

（2）数据表列名：对应于定义 2.2 中数据表的列名；

（3）数据表内容：是数据库中数据表的结构信息；

（4）SQL 语句：符合图 2.8 所示格式的 SQL 句子；

（5）深度学习模型的标注：图 2.8 中包含“#”的部分是预测过程中需要填写的部分，通过分解训练数据集的 SQL 语句得到相应的标注。

本文所做的工作是面向中文的 NL2SQL，该工作把训练数据集中的自然语言查询语句和对应的数据库表结构信息共同作为深度学习模型的输入，并将 SQL 语句的解析数据作为有监督学习的标注来完成两个子任务模型的训练，即列值抽取模型和条件值抽取模型。训练模型之后，通过输入自然语言查询及相应的数据表，合并两个子任务模型的预测结果，即可得到图 2.8 所示的完整 SQL 语句。

本文在处理 NL2SQL 任务时主要分为三个步骤：首先训练列值抽取模型；其次训练条件值抽取模型；最后拼接列值抽取模型和条件值抽取模型的结果。具体步骤讲述如下。

（1）训练列值抽取模型

列值抽取模型是生成图 2.8 中除 VALUE 之外的所有 SQL 语句内容。在此步骤中基于预训练语义模型建立多个分类模型，每个分类模型负责预测 SQL 语句对应部分的内容，如 SELECT 语句中列值部分。本文将自然语言问句与数据表列名同时编码为列值抽取模型的输入，查询语句对应的 SQL 语句作为训练模型的标签，最终得到训练完成的列值抽取模型。

（2）训练条件值抽取模型

训练条件值抽取模型目的是生成图 2.8 中的 VALUE 值。在此步骤，本文基于预训练语言模型构建深度学习分类模型。本文将自然语言查询语句、数据库表名和列名信息作为输入，结合列值抽取模型 WHERE 语句列值预测结果得到条件值，将数据集中 SQL 语句作为真实数据，最终得到本文的条件值抽取模型。

（3）拼接模型结果

拼接模型结果是将上述列值抽取模型的预测结果和条件值抽取模型的预测结果进行组合，组合规则是按照 SQL 语句的语法特征，输出结果则如图 2.8 所示，是自然查询语句对应的完整 SQL 语句。由于条件值的预测需要依赖列值抽取模型的预测结果，所以条件值抽取模型的输入含有列值抽取模型预测结果中条件值的列名，因此，条件值抽取模型必须等到列值抽取模型预测完毕之后才可以进行预测。

2.4 本章小结

本章主要介绍与中文自然语言转结构化查询语句相关的问题分析。首先介绍了面向中文 NL2SQL 的技术具有广泛应用的范围，具体通过目前所使用的落地应用说明该技术在相关行业的重要性，同时引出相关研究与实际应用之间的差距，说明本文所做研究工作的必要性。然后详细介绍了 NL2SQL 相关工作，主要是对基于深度学习的 SQLova 模型和 X-SQL 模型所使用的方法进行分析，两个模型均通过预训练模型提高模型性能并取得出色实验成果。接着对上述两个英文 NL2SQL 模型面向中文进行实验，分析实验结果得到面向中文数据时英文 NL2SQL 模型准确率会有所降低的结论，继而分析了本文中如何基于现有工作解决中文 NL2SQL 的问题。本章最后对中文 NL2SQL 问题的相关概念进行定义，同时提出本文解决中文 NL2SQL 问题所用的研究方法与流程。

3 中文自然语言转结构化查询语句方法

现有自然语言转 SQL 语句（NL2SQL）工作少有关注数据库表中的列名是否被重复使用的情况，在生成 SQL 语句过程中也未能充分利用数据库表信息。本章基于现有研究提出了改进的 NL2SQL 算法，该算法提出使用列值抽取模型和条件值抽取模型来完成中文 NL2SQL 任务，下面主要对上述方法和模型进行介绍。

3.1 任务介绍

本文借鉴 SQL 语句的分割思想，利用分割后每个子语句间的关联将中文 NL2SQL 任务分割成填充槽值的若干子任务。此种思路根据槽值填充任务的特征，将其划分为两个子问题，从而极大地减少了 NL2SQL 问题的工作量，有效提高了 SQL 语句的生成准确率。下面举例说明 NL2SQL 数据集中数据表的数据，其内容如表 3.1 所示。

表 3.1 NL2SQL 数据实例

公司	所属国家	17 年支出	18 年支出	19 年支出
三星	韩国	24 232	22 620	18 000
英特尔	美国	11 778	15 500	13 500
海力士	韩国	8 091	12 800	10 000
台积电	中国台湾	10 846	10 250	10 000
镁光	美国	6 475	9 960	9 500

假设此时自然语言查询语句为“三星和英特尔在 17 年的时候一共支出多少啊”，根据本文第二章图 2.8 中的 SQL 语句功能表可以得到，SELECT 语句中的 AGG 段与聚合操作“SUM”相对应，“17 年支出”是 COLUMN 需要预测的列名，而在问句中的信息则表明，WHERE 语句中包含两个 COLUMN，且被包含的两个 COLUMN 都对应于“公司”，SQL 语句中条件的列名与条件值之间的运算符为“=”，条件之间的 COND_OP 对应连接符“OR”，条件值 VAL 分别对应于为“三星”和“英特尔”，上述信息都包含在数据表内容之中。最终的 SQL 语句应该是“SELECT SUM(17 年支出) WHERE 公司 = 三星 OR 公司 = 英特尔”。

本文在生成 SQL 语句过程中，会分析 SQL 语句中所有子语句的内容，并将分析后的数据以 JSON 格式表示，该结果在模型训练过程中用作标签值。分析的第一步是把 SQL 语句分解为两个子语句，分别是 SELECT 语句和 WHERE 语句。第二步是对每个子语句进行结构分析，SQL 语句完成解析的结果用“sql”对应的值表示，如图 3.1 所示。


```
{
  "table_id" : "4d29d0513aaalle9b911f40f24344a08",
  "question" : "三星和英特尔在17年的时候一共支出多少啊",
  "question_token" : ["三","星","和","英","特","尔","在","1","7","年","的","时","候","一","共","支","出","多
  "sql" : {
  "select_num" : 1,
  "rpt" : 1,
  "agg" : [5],
  "where_num" : 2,
  "cond_op" : 2,
  "sel" : [2],
  "conds" : [
      [0, 2, "三星"],
      [0, 2, "英特尔"]
    ]
  }
  "QEv" : [1, 3, 0, 1, 2, 3, 0, 4, 4, 4, 0, 0, 0, 0, 0, 4, 4, 0, 0, 0],
  "HEv" : [2, 0, 1, 0, 0]
}
```


图 3.1　SQL 语句解析结果

如图 3.1 所示，“rpt”标签代表最终生成的 SQL 语句中是否存在被复用的列名；“select_num”表示 SELECT 语句选择列的总数；“agg”代表 SELECT 语句对选中的列所做的聚合操作；“where_num”表示 WHERE 语句中选择数据表列名总数；“sel”表示 SELECT 语句选择的列；“conds”可以看做列表，列表里面存放若干三元组，代表 WHERE 语句中一系列对条件列与条件值比较的操作，每个操作都是一个条件列、条件运算符和条件值构成的三元组；“cond_op”则代表 conds 中各条件之间的并列关系。上面提到的所有标签都是生成 SQL 语句时需要赋值的内容，并以某种方式相互关联，例如，SELECT 语句中选中的列名会影响其中出现的聚合操作符。

3.2 外部特征向量编码

由于现有模型没有使用表内容信息，本文提出的算法如下所述。在 BERT 模型的基础上，使用表格内容信息解决中文 NL2SQL 问题。根据观察到的数据属性，例如表 3.1 中，表内容“三星”和“英特尔”的内容对应于等待查询的自然语言问句中的内容，而表头中“17 年支出”同样对应于查询语句中的内容。本文为了把数据库中数据内容，表头内容整合到 NL2SQL 模型中，独立设计了两个外部特征向量。

在本文中，数据库表内容特征向量（QE_v）是基于数据库表格的数据以及查询语句中的字符串构造的，因此QE_v向量的长度与自然语言查询语句的长度一样；数据库表头特征向量（HE_v）则是基于数据库表格的所有表头字段和自然语言查询语句中字符串的匹配信息来构造的。下面说明数据库表内容向量和数据库表头向量编码的详细过程。

3.2.1 数据库表内容向量的编码

本文对数据库表内容向量（QE_v向量）的构造过程如下：首先，定义与自然语言问题长度相同的QE_v向量；其次，对数据表格中所有的单元格内容进行了遍历，如果在遍历过程中匹配到与自然语言查询语句中内容相同的单元格内容，就在QE_v向量的匹配成功字符串的开头和结尾部位分别赋值为 1 和 3，而位于开头和结尾之间的都赋值为 2；最后将数据表中所有表头遍历一遍，将QE_v向量中与列名匹配成功的位置赋值为 4。

本文在对数据库表内容进行编码时，数据库表格内容信息在自然语言查询语句开始位置的初始值设置为 1，中间位置则赋值为 2，结束位置赋值为 3，数值 4 则表示自然语言查询语句中该位置是数据库表的列名。

3.2.2 数据库表头向量的编码

本文对数据库表头向量（HE_v向量）的构造过程如下：首先，对HE_v向量进行定义；其次，对数据库表中所有的列进行遍历，如果在遍历过程中匹配到列名和自然语言查询问句中的内容，就在HE_v向量的对应位置设置为 1；最后对数据库表中的所有的单元格内容进行遍历，在自然语言查询问句中匹配到单元格内容的位置赋值为 2。

本文在对数据库表头向量进行编码时，使用数值表示匹配结果，其中数值 1 表示该位置的数据库列名在自然语言查询问句匹配成功，数值 0 表示未匹配成功，数值 2 则表示该列的数据库内容与自然语言查询问句中的内容匹配成功。

3.3 中文自然语言转 SQL 语句模型

上文已对数据库表内容向量和数据库表头向量的编码过程进行详细说明，下面对本文 NL2SQL 模型的结构进行介绍，并讲述建模过程。本文使用两个模型完成 NL2SQL 任务，其一是生成 SQL 语句选择的所有列名、聚合操作符、条件连接符和运算符等内容，由本文的列值抽取模型完成；其二是生成 WHERE 语句中条件值，由本文提出的条件值抽取模型完成。最终可以得到本文 NL2SQL 模型的整体模型结构如图 3.2 所示。

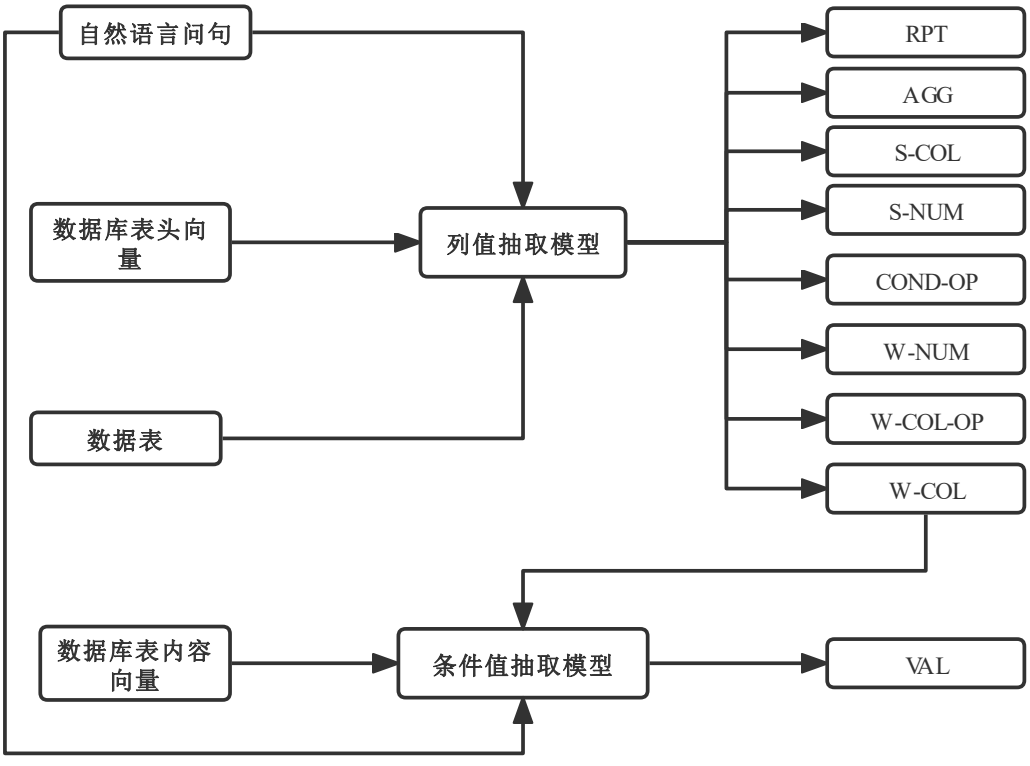

图 3.2 模型结构简图

从本文的NL2SQL模型结构图中可以看出，列值抽取模型被分为了 8 个子任务，上述子任务预测结果的选值范围都是有限的。比如 AGG 的候选值只有数据库中聚合操作类型，所以列值抽取模型的子任务全都是分类问题。条件值抽取模型与列值抽取

模型中的子任务有所不同，其作用是进行条件值预测。本文使用抽取式方式生成 SQL 语句中的条件值。本文提出列值抽取模型加条件值抽取模型之后，使上述两个模型分别处理 SQL 语句生成过程中不同的任务，并整合其结果得到最终的 SQL 语句。

下面对图 3.2 中各个子任务模块从上到下依次进行说明。其中，RPT 表示 SQL 语句的 WHERE 语句是否有重复使用的数据列名；AGG 表示 SQL 语句中的聚合操作；S-COL 则表示 SQL 语句中 SELECT 语句选中的数据库表中的列名；S-NUM 表示选择语句选中数据库表中列的总数；COND-OP 在 SQL 语句中是描述 WHERE 语句各种条件之间的关系的；W-NUM 与 S-NUM 类似，代表了 WHERE 语句选择的数据库表中列名的数量；W-COL-OP 表示 WHERE 语句中的操作符；W-COL 表示 WHERE 部分所选的数据表列名；最后 VAL 则表示 SQL 语句中出现的条件值。

3.3.1 列值抽取模型

本文中列值抽取模型是多任务分类模型，该模型包含多个任务共享的底层编码，该模型主要用于对查询出的 SQL 语句中除条件值以外的其他内容进行预测。每处需要预测的内容都是分类任务，列值抽取模型在进行某个子任务时会生成相应的多维向量，该向量的维度代表了某任务的取值种类总数，而向量中的每个数值代表了某任务取此对应值的几率。为了更好的表示此种向量的映射关系，在表 3.2 中展示了其映射的结果。

表 3.2 中的 RPT 共有两个类别，其取值可以是 0 或是 1。该模块用来表示 SQL 语句中 WHERE 语句是否有列名被重复使用，以帮助模型更精准地预测数据库表中出现在 WHERE 语句中的列名。在列值抽取模型中，S-NUM 和 W-NUM 的预测值都是数字类型，用来表示选中列的数量。本文对 SQL 语句中选中的列数进行了统计分析，每条 SQL 语句对应的列数均在 4 个以下，所以本文将上述预测值种类的最大值都设定为 3。AGG 表示 SQL 语句中对某列聚合操作的类型，与处理后的数据中“agg”相对应；W-COL-OP 表示 SQL 语句中的运算符；COND-OP 表示 SQL 语句各种条件之间的关系。其中 W-COL 和 W-COL-OP 以及条件值共同组成“conds”中的元素。上面提到的内容可以组成完整 SQL 语句结构，而且语句结构不会改变。但是在 SELECT 语句和 WHERE 语句中可以选择的数据库列名是会随着数据库表发生改变

表 3.2　子任务映射关系表

RPT	类别
0	0
1	1
S-NUM	类别
0	0
1	1
2	2
3	3
AGG	类别
NONE	0
AVG	1
MAX	2
MIN	3
COUNT	4
SUM	5
W-NUM	类别
0	0
1	1
2	2
3	3
W-COL-OP	类别
>	0
<	1
=	2
!=	3
>=	4
<=	5
COND-OP	类别
NONE	0
AND	1
OR	2

的，数据库表名又随着给定的自然语言查询语句而改变。在表 3.1 中的实例中，列名取值范围为{“公司”，“所属国家”，“17 年支出”，“18 年支出”，“19 年支出”}。

本文对 SQL 语句中 WHERE 语句的列名选择情况进行了分析，结果表明 WHERE 子句选中多个列名时，会出现同一列名出现多次的情况。本文提出利用选中列名的数量来确定具体的标记的方法，经过对数据集中数据的分析发现本文工作中 SQL 语句最多包含三个列名，所以本文在模型中将为每个数据的列名提供了三维向量标签，如表 3.3 所示。

表 3.3　列名 COL 的标签标记方式

WHERE 语句选择列名的数量	标签
0	[0, 0, 0]
1	[1, 0, 0]
2	[1, 1, 0]
3	[1, 1, 1]

列值抽取模型基于上述标记方式对图 3.1 中所示例的 WHERE 语句选择的数据列进行标签标记，数据列对应的标记如表 3.4 所示。

表 3.4　WHERE 语句中的列名标记结果

列名	公司	所属国家	17 年支出	18 年支出	19 年支出
W-COL	[1, 1, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]	[0, 0, 0]

在模型的推理过程中，如果 SQL 语句没有出现一个及以上相同的列名，RPT 预测为 0，将所有列名的预测标签首个元素进行从大到小的排序操作，再提取出与 W-NUM 数量相等的数据库列名；SQL 语句中出现重复使用的列名（即 RPT 预测值为 1）时，不再只对预测标签的首个元素进行排序，而是对所有列名预测标签的所有元素进行从大到小的排序，同样按顺序取出 W-NUM 个数据库列名。

图 3.3 是列值抽取模型的骨架结构图，该模型包括三大模块：特征表示编码、特征增强和分类模型。下面详细说明该模型原理。

首先，利用 Chinese-BERT-wwm 模型对输入的自然语言查询文本和数据库表的列名的语法和语义特征进行提取，再由 Chinese-BERT-wwm 模型输出其编码向量。

其次，将分隔符[SEP]的输出向量与完成编码的数据库表头向量（即HE_v向量）相结合，并将两个向量合并后的结果进行特征增强，而特征增强部分则是进一步加强数据库列名信息的特征，并基于特征增强的向量表示构造了八个子任务的分类模型，分别对应于图 3.2 中列值抽取模型的子模型。最后，通过与表 3.2 中的各标签相互对应，子任务模型的输出分别组成需要生成的 SQL 语句中的不同语句。

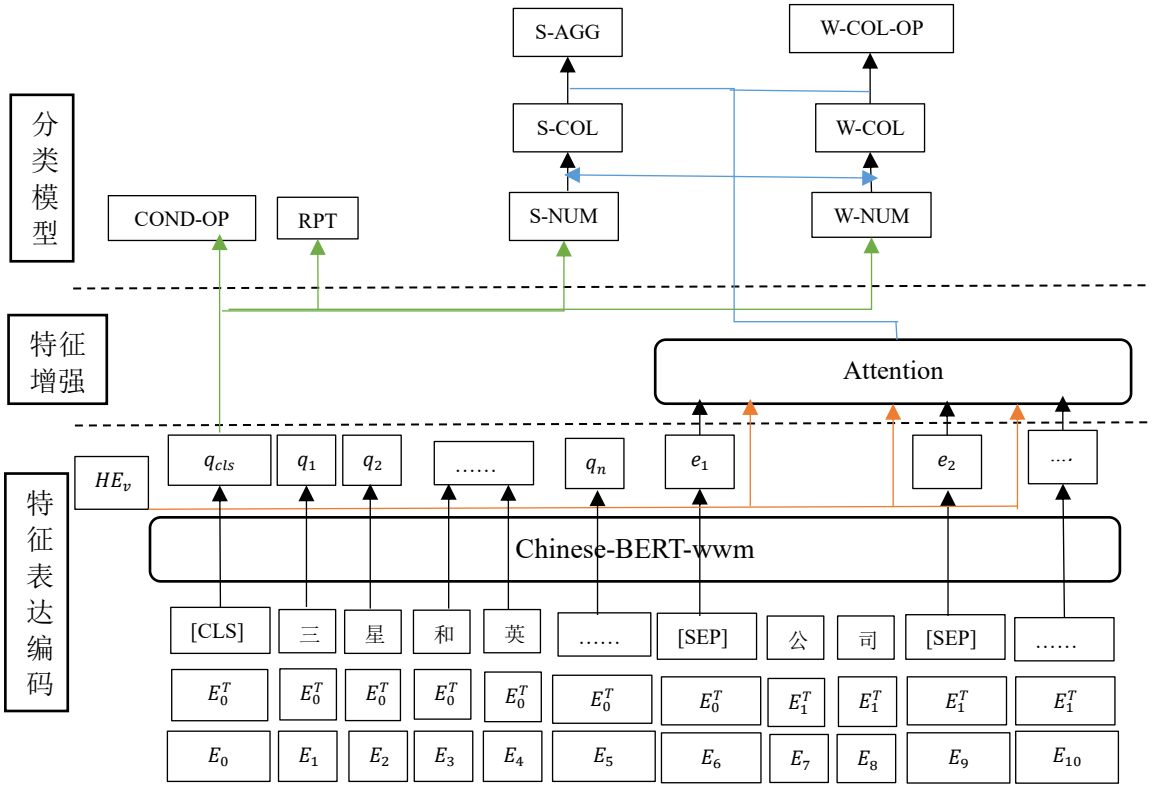

图 3.3　列值抽取模型结构

从图 3.3 中的特征表示编码层可以看出，本文对问句信息和列名信息进行特征编码时使用了原始 BERT 中的方法，仍然将 Token Embeddings、Segment Embeddings 和 Position Embeddings 三种类型信息作为输入信息。其中E_0^T向量代表自然语言问句信息，数据库表中列的名称用E_1^T向量。

（1）特征表达编码

本文采用在预训练模型进行特征提取的基础上进行模型微调的方法，将自然语言查询语句和数据库表全部列名进行拼接，全部作为预训练模型的输入序列。经过上述对输入数据所做的处理，表 3.1 的数据实例在预训练模型中编码层的问题标记、列

标记和输入标记如图 3.4 所示。

本文的特征表达编码层需要对自然语言查询问句和数据表的列名全都进行编码操作，所以在图 3.4 中，输入标记包含问题标记和所有列标记，表示的是预训练模型完整的输入序列，自然语言查询问句定义如公式(2.1)所示，数据表列名的定义如公式(2.2)所示，所以在本文中，预训练模型的输入序列可以描述为公式(3.1)所示。

$$Input = concat(CLS, question, SEP, h_1, SEP, h_2, SEP, \dots) \tag{3.1}$$

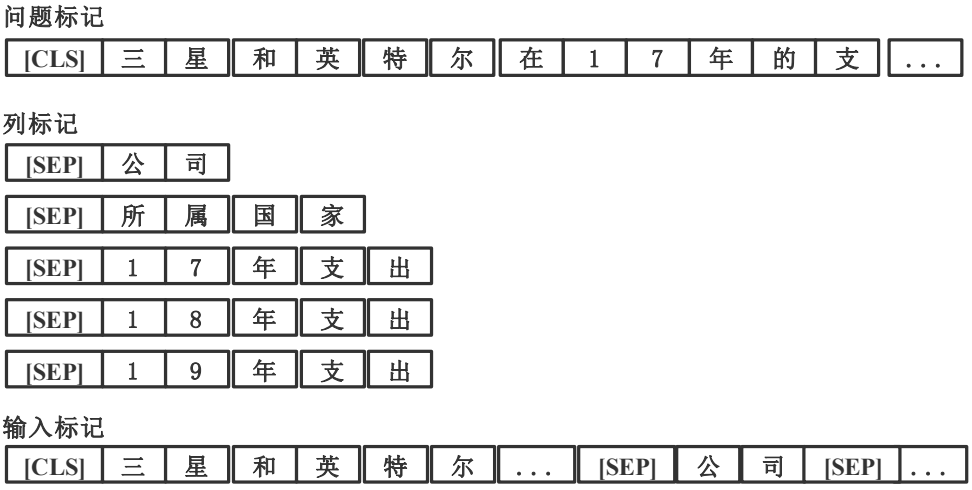

图 3.4　预训练模型的输入处理

公式(3.1) 的 CLS 表示预训练模型输入的开始，SEP 表示输入序列中自然语言查询问句的结束，或是表示数据库中列名的结束，模型输出端会将所有字符的语义信息都输出，在本文中只取出其中两个向量作为下一层的输入，分别是含有全部序列信息的向量q_{cls}和每个[SEP]的输出向量e_i。在整个输出信息中，选择所有的[SEP]输出向量组成新向量H_v，并将其表示为$H_v = \{e_1, e_2, \dots, e_n\}$，其中 n 表示数据库表的列数。其公式可表示为公式(3.2)所示，$Model$表示预训练模型。

$$\{q_{cls}, H_v\} = Model(Input) \tag{3.2}$$

（2）特征增强

预训练模型得到所有字符的向量之后，本文将对数据库表中所有列名和问句中的出现列名做特征增强，做法是将之前编码的数据库表头向量（HE_v）和自然语言查询问句中出现的列名的语义向量H_v进行组合，得到的结果作为注意力模型的输入。NL2SQL 任务中使用预训练模型对问句进行编码的方法采用注意力机制，计算公式

如公式(3.3)所示。计算结果H_h是数据库表在自然语言查询问句中出现的列名的语义向量经过 Attention 之后的向量。

$$H_h = Attention(Z, Z, Z) \tag{3.3}$$

公式(3.3)中的Z分别表示自注意力机制中的 Query、Key 和 Value，Z的计算方法如公式(3.4)所示，其中W_h在训练过程中是可训练的参数。

$$Z = H_v + HE_v \cdot W_h \tag{3.4}$$

经过上述的计算过程，该层对列的信息特征做了增强处理，最终取出序列信息的特征向量q_{cls}和公式(3.3)的结果作为分类模型的输入。

（3）分类模型

本文将 NL2SQL 任务划分为不同的分类子任务，而每个子任务的类别数目又是有限的且低于模型中向量的维度，所以本文对所有的子任务都采用 Dense 层，将前面的向量维度降维成为具有子任务的取值类别数目维度的向量。

经过上一层之后，可以根据各个子任务的特征向量建立对应的子任务模型，本文在对不同子任务处理时使用的特征向量也不同，其中数据库与自然语言查询问句中对应的列名的特征表达式H_h用作处理下面四个子任务：S-COL，AGG，W-COL 和 W-COL-OP。为便于后续分类模型的说明，本文给出关于矩阵U的分类概率分布的计算公式，包括公式(3.5)和公式(3.6)。

$$P_c(U) = softmax(Wtanh(U)) \tag{3.5}$$

$$tanh\,(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.6}$$

本文中各个子任务类别的数量确定了分类概率的维度，$softmax$ 函数用作数值的归一化处理，W在公式中代表了可训练的参数，$tanh$ 是最常见的激活函数之一。$softmax$ 函数的计算公式如公式(3.7)和公式(3.8)所示。

$$y_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}, i = 1,2,\dots,N \tag{3.7}$$

$$\mathrm{t}\sum_{i=1}^{N} y_i = 1 \tag{3.8}$$

其中，式中的y_i代表了概率值的计算，i表示的是分类子任务中第i类（也是输出数据向量的第i维度）；公式中的N表示各个子任务中的分类类别数，同时也是上述向量的

维度；Z_j则代表输入向量第j维度的数值。

接下来对各个分类模型所用到的计算公式进行定义，其中W_i是模型训练过程中可学习的参数。

① RPT 标志

子任务 RPT 模块的预测集是{0, 1}，用来预测 SQL 语句中是否会出现重复使用的数据库列名。本文利用序列信息q_{cls}向量作为输入来构建 RPT 模块的二分类模型，RPT 模块的概率记为p_1，其分布公式如公式(3.9)所示，其中，$sigmoid$函数是可用于二分类模型输出层的激活函数，其计算公式如公式(3.10)所示。

$$p_1 = sigmoid(W_1 q_{cls}) \tag{3.9}$$

$$sigmoid(z) = \frac{1}{1 + e^{-z}} \tag{3.10}$$

② S-NUM 预测

子任务 S-NUM 模块的作用是进行 SQL 语句中 SELECT 语句选中数据库列名数量的预测，经统计分析本文工作中使用的数据集之后，将该模块的预测集设置为{0, 1, 2, 3}，S-NUM 分类概率分布p_2的计算公式如公式(3.11)所示。

$$p_2 = P_c(W_2 q_{cls}) \tag{3.11}$$

③ S-COL 预测

S-COL 模块是生成 SELECT 语句中选择的数据库列名，但该模块可以在 S-NUM 模块的基础上进行预测，所以 S-COL 模块是在 S-NUM 的结果之上进行预测。由于该模块预测的是数据库列名，所以预测的集合是由数据库中所有列名组合而成。该模块的分类概率分布记为p_3，其计算过程表示为公式(3.12)。

$$p_3 = P_c(W_3 H_v + W_4 H_h) \tag{3.12}$$

④ AGG 预测

S-COL 模块预测 SQL 语句选定的数据库列名后，将使用 AGG 模块预测 SQL 语句中对列名使用的聚合操作。其分类概率分布p_4的计算如公式(3.13)所示，该公式的计算结果表示选择的数据库列名为数据库表中列名为h_i的输出概率。

$$p_4(h_i) = P_c(W_5 H_v + W_6 H_{h_i}) \tag{3.13}$$

⑤ W-NUM 预测

W-NUM 模块用于预测 WHERE 语句选中列名的总数，类比 S-NUM，W-NUM 任务的预测集也是为{0, 1, 2, 3}，该模块的分类概率分布p_5的计算过程如公式(3.14)所示。

$$p_5 = P_c(W_7 q_{cls}) \tag{3.14}$$

⑥ COND-OP 预测

在 SQL 语句中可能存在多个条件，COND-OP 模块用来预测此种情况下不同条件之间的关系，预测集为{None，“AND”，“OR”}，该模块的分类概率分布p_6的计算过程如公式(3.15)所示。

$$p_6 = P_c(W_8 q_{cls}) \tag{3.15}$$

⑦ W-COL 预测

W-COL 模块的预测同样可以在 W-SUM 的预测基础上进行预测，预测集与 S-COL 相同，也是数据库表中列名与自然语言查询问句中相匹配的所有列名。该模块的分类概率公式p_7表示为公式(3.16)所示。

$$p_7 = P_c(W_9 H_v + W_{10} H_h) \tag{3.16}$$

⑧ W-COL-OP 预测

SQL 语句中的 WHERE 语句选中列之后，该列与对应的条件值之间的运算符也需要被预测，W-COL-OP 模块的作用是预测运算符，其预测的范围是{“>”，“<”，“=”，“!=”，“>=”，“<=”}中任意运算符。该模块的分类概率分布记为$p_8(h_i)$，如公式(3.17)所示，$p_8(h_i)$表示选择的数据库列名为数据库表中列名为$\mathrm{h_i}$的输出概率。

$$p_8(h_i) = P_c(W_{11} H_v + W_{12} H_{h_i}) \tag{3.17}$$

由于列值抽取模型的子任务预测范围都是有限且已知的，所以本文在训练列值抽取模型时对每个分类子任务进行并行训练。但是模块预测无法并行执行，其原因是模块之间存在依赖关系，比如 W-COL 模块依赖于 W-NUM 模块的预测结果，所以 W-NUM 模块预测完成后才可预测 W-COL 模块。

3.3.2 条件值抽取模型

在列值抽取模型生成各个子任务预测值之后，完整的 SQL 语句还需要获取 WHERE 语句中的条件值来完成。条件值是需要与 WHERE 语句中选择的条件列进

行拼接的，所以条件值抽取模型将自然语言问句与 W-COL 所有预测结果建立关系。例如，自然语言问句为“你知道财务部需要招多少名会计员吗”，若 W-COL 的预测结果的列名为“部门”和“职位”，详细的拼接过程如图 3.5 所示。

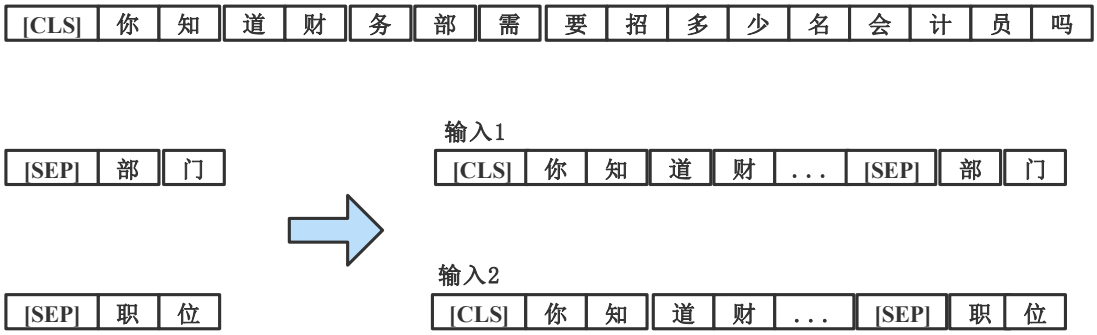

图 3.5　自然语言问句与候选列的拼接过程

本文做法是将 W-COL 子模型预测结果都建立上述关系，并作为 Chinese-BERT-wwm 模型的输入对其进行编码，编码结果与数据库表头向量融合来预测此列名与自然语言问句中条件值的对应关系。将输入的自然语言查询语句分割为单个字符，每个字符都生成相应的标志，标志为 0 或 1，分别表示条件值在该位置或不在该位置，具体如图 3.6 所示。

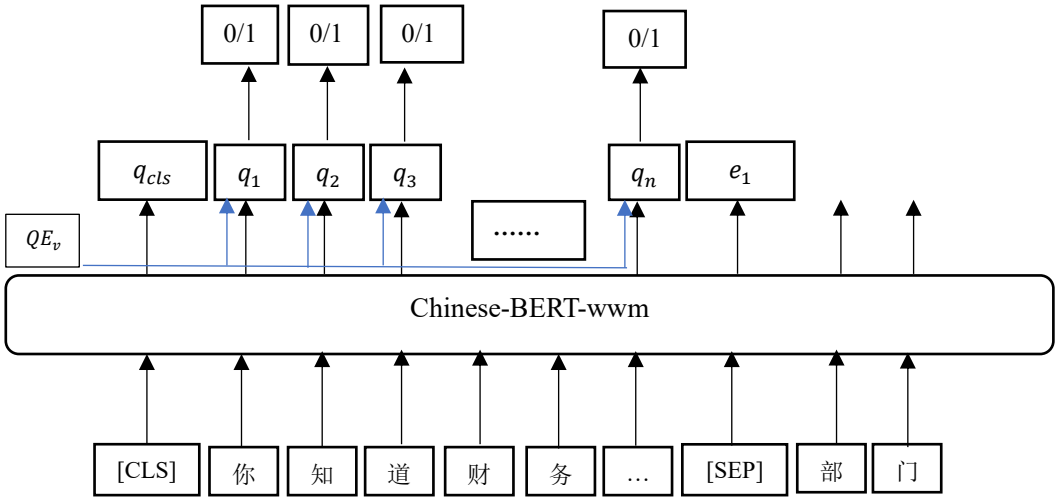

图 3.6　基于分类标签的条件值抽取结构简图

在预测条件值时，自然语言查询问句中通常会包含 SQL 语句的条件值，因此，为能找出 SQL 语句条件值，需要找到条件值在该问句中的位置。在公式(3.18)中，其

结果表示的是问句中第i个字符被预测为条件值的概率；q_i表示的是问句经过Chinese-BERT-wwm后第i个字符的特征向量；QE_{V_i}表示的是数据库表内容向量QE_V的第i个位置的元素；W_e表示模型中通过学习可更新的参数。

$$p_9 = sigmoid\left(W_{13}(q_i + QE_{V_i} \cdot W_e)\right) \tag{3.18}$$

本文使用的标记方法，SQL 语句中对条件值抽取的计算方法如公式(3.19)所示。

$$\{p_{9_1}, p_{9_2} \dots p_{9_i} \dots p_{9_n}\} = Model\_val(Input_i, QE_v) \tag{3.19}$$

其中，p_{9_i}表示的是自然语言问句中第i个字符属于该值的概率；条件值抽取模型为$Model_val$。

3.3.3 损失函数的设计

损失函数（Loss Function）用来评估模型的预测值和真实值之间的差异，可以评价模型在样本上的表现，是深度学习参数优化至关重要的部分，最小化损失函数是模型训练过程中收敛的重要步骤。本文在训练列值抽取模型和条件值抽取模型时，对各子任务独自使用损失函数，并将其损失函数值相加作为最终的损失函数，模型训练时会使各子任务的损失函数之和达到最小，从而来使模型达到最优的效果。本文各子任务都是分类问题，所以本文采用适用于分类问题的交叉熵损失函数，其计算过程如公式(3.20)所示。

$$loss(P(x), y) = -\sum_{i=1}^{N}\sum_{j=1}^{K}\left(y_{ij} log_2 P(x_{ij}) + (1 - y_{ij}) log_2\left(1 - P(x_{ij})\right)\right) \tag{3.20}$$

公式(3.20)表示的是某个子任务模型的损失函数，其中N表示该子任务中所有样本数；K表示该子任务进行分类的数量；y_{ij}代表第i个样本第j类标签的参考值，表示该值是否为真，取值为0或1，而$P(x_{ij})$则表示预测y_{ij}的概率值。本文在计算模型损失函数时是将模型的各个子任务模型的损失函数相加成为模型的总损失函数。

但是，在使用S-COL、W-COL和VAL子模块时，在数据表中存在着若干列名，并且在每一个列名下面都对应若干列值，而在列名中，仅有极少数的列名和少量列值是训练正样本，真实SQL语句中未出现的都是负样本，此种情况会造成正负两个样本的失衡，从而影响到模型的学习。针对上述问题，本文采用了带有权重的交叉熵损失函数，并给出了相应的计算公式，如公式(3.21)和公式(3.22)所示。其中，公式(3.21)

表示 S-COL 模块的损失函数计算公式，也可用于 W-COL 模块的损失函数计算，而公式(3.22)是 VAL 模块的损失函数计算。本文综合考虑经验值和正负样本比例情况将正负样本的超参数α和β分别取值为 3 和 5。

$$loss(P(h),y) = -\sum_{i=1}^{N}\sum_{j=1}^{K}\left(\alpha y_{ij} log_2 P(h_{ij}) + (1-y_{ij}) log_2\left(1-P(h_{ij})\right)\right) \quad (3.21)$$

$$loss(P(val),y) = -\sum_{i=1}^{N}\sum_{j=1}^{K}\left(\beta y_{ij} log_2 P(val_{ij}) + (1-y_{ij}) log_2\left(1-P(val_{ij})\right)\right) \quad (3.22)$$

3.4 本章小结

本章主要介绍本文解决中文 NL2SQL 问题所使用的建模方法。首先详细介绍了本文对中文自然语言转 SQL 语句的任务分析，说明数据库信息和自然语言问句在本文工作中对 SQL 语句的生成起到关键作用。其次提出本文在建模时高效利用表内容信息的方法，通过数据表和自然语言问句相结合构造两个外部特征向量，分别是数据库表内容向量和数据库表头向量，并给出其详细编码过程。接着对本文所采用的方法及整体模型结构进行介绍，结合预训练语言模型和深度学习分类模型建立列值抽取模型和条件值抽取模型，其中列值抽取模型通过八个分类子任务模型完成 SQL 语句中除条件值以外所有内容的预测，条件值抽取模型通过结合列值抽取模型的结果建立二分类模型完成条件值的预测，并给出详细建模过程。本文所建立模型通过预训练语言模型进行特征编码并进行特征增强，以此提高模型对自然语言问句和数据库信息的识别能力，提高模型预测的准确度。本章在最后介绍了本文模型针对不同问题如何设计交叉熵损失函数。

4 实验结果与分析

为评估本文所提方法是否有效，本章首先介绍本文实验数据以及数据处理工作，说明模型训练时的实验设置及其环境，其次选择具有代表性的模型和本文模型进行对比实验，最后得到实验结果并根据已有评价标准验证本文模型的性能。

4.1 数据预处理

本文实验选择中文 NL2SQL 数据集作为训练数据，该数据集的实例数据格式如图 4.1 所示。中文 NL2SQL 数据集具有其他数据集所未有的特点，即由自然语言生成 SQL 语句时需要考虑数据库表结构信息。本文所提模型在生成 SQL 语句时必须依赖该数据集中相应表格的 id（即 table_id）及自然语言查询问句（即 question），所以在训练模型时需要将数据库表结构信息和自然语言语句信息同时作为模型的输入。


```
{
  "table_id" : "a1b2c3d4",
  "question" : "世茂茂悦府新盘容积率大于1，它的套均面积是多少",
  "sql" : {
    "sel" : [7],
    "agg" : [0],
    "cond_op" : 0,
    "conds" : [
      [1, 2, "世茂茂悦府"],
      [6, 0, "1"]
    ]
  }
}
```


图 4.1 中文 NL2SQL 数据样例

由于中文语言具有丰富的表达方式，中文自然语言问句包含的信息可能不会出现在数据库表格中，例如自然语言问句中出现的“鹅厂”在数据库表中对应的数据是“腾讯”，因此在训练 NL2SQL 模型时需要将语句和数据库表结合起来。

另外，在中文 NL2SQL 数据集中有些问句存在有关年份、数字、单位和日期等信息，但中文表述方式较为丰富，比如会出现“哪个公司 18 年 12 月 28 号成立？”此种问句中日期形式在实验数据中应该是“2018/12/28”。为了解决数据库中存储的

数据和问句中表述不一致问题，本文基于正则表达式对自然语言问句进行规则修正。

4.2 实验设置与实验环境

本文实验在模型训练和预测时均使用实验室的计算机进行，其中使用的数据集和实验过程中使用及生成的所有文件都存放在该计算机。本实验使用的计算机配置如表 4.1 所示。

表 4.1 实验计算机配置

实验环境	说明
机器型号	B85-PRO GAMER
显卡	NVIDIA Corporation GP102 [GeForce GTX 1080 Ti]
内存	32G
操作系统	Linux
编程语言	Python3.6
BERT 模型权重	哈工大讯飞联合实验室 Chinese-BERT-wwm
深度学习框架	TensorFlow

本文实验使用 Chinese-BERT-wwm 作为预训练模型，该模型是 BERT 模型的升级版本，其层数为 12 层，输入的维度为 768 维。中文 NL2SQL 数据集的训练数据样本最大长度小于 BERT 可以处理的最大长度，所以本文直接将训练数据样本最大长度作为 Chinese-BERT-wwm 的输入长度。

本文实验所得到训练模型时设置的参数如表 4.2 所示。

表 4.2 子模型训练参数设置

模型名称	学习率	批量大小	epoch
列值抽取模型	1e-5	16	30
条件值抽取模型	1e-5	16	30

本文对列值抽取模型训练时，学习率使用 1e-5 到 5e-5 之间，批量大小设置为 8 到 16 之间，模型的 epoch 设置为 30。尝试几次实验之后发现，批量大小为 16，学习率为 1e-5 时，可以得到最优列值抽取模型。

本文在对条件值抽取模型训练时，使用 Adam 作为优化器，学习率使用 1e-5 到 5e-5 之间数值，批量大小为 8 到 16 之间，模型的 epoch 设置为 10。尝试了几次实验

之后发现，当批次大小为 16，学习率为 1e-5 时得到了最好的模型效果。

4.3 评价标准与对比模型

4.3.1 评价标准

本文使用的评价标准计算公式如公式(4.1)和公式(4.2)所示，可以看出准确率的计算方法使用的是计分的方法，每条样本数据经过模型预测后的值与真实值相同时计分为 1，不同则计分为 0，最终对所有样本数据的得分之和求平均值得到准确率。在公式(4.1)中，y'代表模型预测的结果，y代表真实的 SQL 语句，$Score$则代表该样本经过预测的得分。在公式(4.2)中，其中$Score_i$表示第i个样本的得分，N则表示样本数据数量，Acc则代表求出的准确率。

$$Score = \begin{cases} 1, y' = y \\ 0, y' \neq y \end{cases} \tag{4.1}$$

$$Acc = \frac{1}{N}\sum_{i=1}^{N} Score_i \tag{4.2}$$

NL2SQL 任务的效果评估没有标准统一的度量，目前广泛使用的有以下两种方式。其一是执行准确率（Execution Accuracy），即计算通过 NL2SQL 模型生成的 SQL 语句执行结果正确的数量在数据集中的比例。其二是逻辑形式准确率（Logical Form Accuracy），即计算 NL2SQL 模型生成的 SQL 语句与数据集中 SQL 语句完全匹配的数量在数据集中的比例。基于Acc的计算方法，本文针对中文 NL2SQL 任务使用三个不同的Acc指标来评估本文模型生成 SQL 语句的效果，分别是逻辑形式准确率、执行准确率和平均准确率（Mean Accuracy）。其中逻辑准确率的计算要求相对比较高，要求模型预测的结果与数据中真实存在的 SQL 语句数据完全一致，否则不计入逻辑准确率的得分。其计算公式和表示形式如公式(4.3)和公式(4.4)所示。

$$Score^{LA} = \begin{cases} 1, SQL' = SQL \\ 0, SQL' \neq SQL \end{cases} \tag{4.3}$$

$$Acc_{LA} = \frac{1}{N}\sum_{i=1}^{N} Score_i^{LA} \tag{4.4}$$

其中，计算结果$Score^{LA}$表示某样本的得分情况，Acc_{LA}表示逻辑准确率，公式(4.3)中的SQL'和SQL分别代表本文模型对自然语言查询问句生成的预测值和真实值的 SQL 语句。

执行准确率与逻辑准确率相比要求有所降低，该准确率在计算过程中，只对模型预测生成的 SQL 语句在执行之后得到的查询结果有要求，如果该结果与数据集中标准 SQL 语句执行的结果相同，则计为一次得分，否则不计入得分。执行准确率的计算公式和表示形式如公式(4.5)和公式(4.6)所示。

$$Score^{EA} = \begin{cases} 1, Y' = Y \\ 0, Y' \neq Y \end{cases} \tag{4.5}$$

$$Acc_{EA} = \frac{1}{N}\sum_{i=1}^{N} Score_i^{EA} \tag{4.6}$$

其中，计算结果$Score^{EA}$表示某样本的得分情况，Acc_{EA}表示执行准确率，公式(4.5)中的Y'和Y分别代表本文模型对自然语言查询问句生成的 SQL 语句执行结果和真实 SQL 语句的执行结果。

平均准确率的目的为了更好的评估模型的性能，结合现有的Acc_{LA}和Acc_{EA}进行模型评估，本文使用平均准确率表示Acc_{LA}和Acc_{EA}的均值，其计算公式和表示形式如公式(4.7)所示。

$$Acc_{MA} = \frac{Acc_{LA} + Acc_{EA}}{2} \tag{4.7}$$

4.3.2 对比模型

本文在实验过程中，选取了 NL2SQL 任务中典型的研究结果进行了对比试验，以验证本文所提模型与算法在中文 NL2SQL 中的应用效果。具体对比模型如下。

Seq2Seq[28]：传统的 NL2SQL 模型，解码器部分的单词表固定，生成的 SQL 语句序列也是固定的。

Seq2SQL[32]：将生成的 SQL 语句分成三个部分，每部分使用不同的方法进行计算，没有利用 SQL 语句的语法结构。

SQLNet[35]：将 SQL 语句分成两部分，每个部分设置槽位进行槽值填充。

TypeSQL[36]：该模型基于 SQLNet 模型，将 NL2SQL 问题转换为插槽填充任务，并使用三个独立模型来预测插槽填充值。

SQLNet+BERT[55]：在实验过程中，通过改进模型，采用 BERT 模型实现自然语言和数据库列名编码。

SQLNet+Chinese-BERT-wwm[60]：该模型与上述实验方式相同，但此条实验是使

用 Chinese-BERT-wwm 模型进行自然语言和数据库列名编码。

SQLova[40]：借鉴了 SQLNet 中对 NL2SQL 任务进行分步解决的思想，但 SQLova 使用 BERT 作为模型的输入表达层，代替了词向量。

X-SQL[56]：使用了 MT-DNN 预训练模型的上下文输出来增强结构模式表示，并结合类型信息来学习用于下游任务的模式表示。

4.4 中文自然语言转 SQL 语句算法实验

4.4.1 列值抽取模型训练

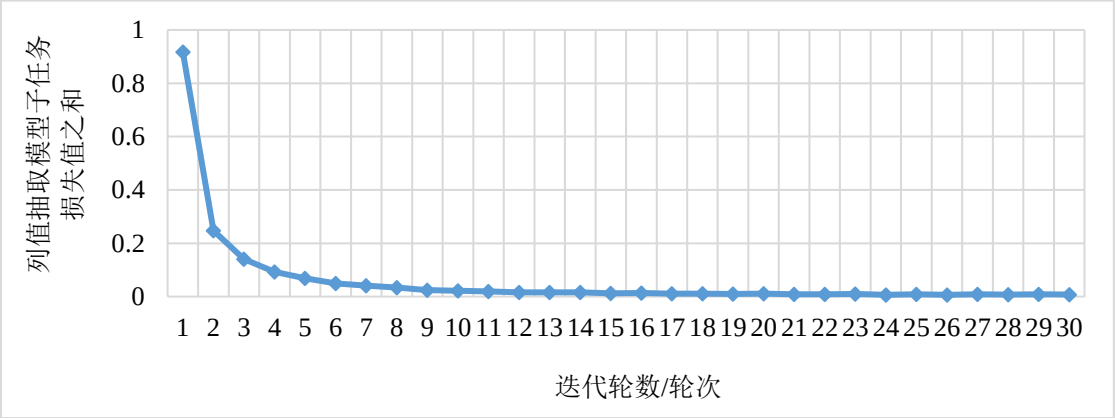

图 4.2 列值抽取模型训练过程中子任务损失值之和变化

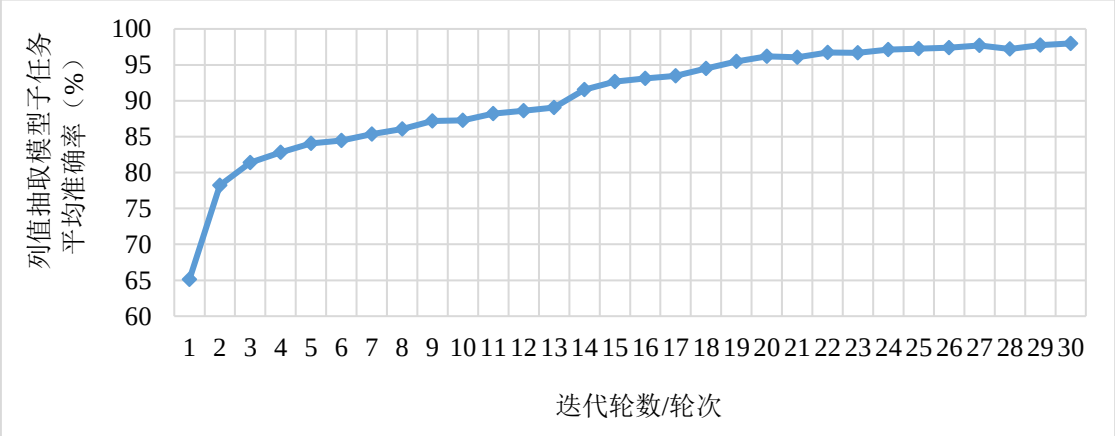

图 4.3 列值抽取模型训练过程中子任务平均准确率变化

本文在列值抽取模型训练过程中，模型的整体损失值和平均准确率随迭代轮数 epoch 变化如图 4.2 和图 4.3 所示，可以看出列值抽取模型在训练过程中损失值的整体下降趋势以及平均准确率的上升趋势，并且最终趋于平稳。

4.4.2 列值抽取模型结果分析

本文提出的中文 NL2SQL 模型中的列值抽取模型（MyModel1）在训练如图 3.2 所示被分为 8 个子任务模型，其中，在对比模型中不存在 RPT 标签子任务，同时 Seq2Seq 模型和 Seq2SQL 模型中也未提到 S-NUM 子任务和 W-NUM 子任务。所以，在中文 NL2SQL 数据集的验证集上，本文的列值抽取模型和对比模型中对各个子任务上的逻辑准确率如表 4.3 所示。

表 4.3 不同模型在各子任务的逻辑准确率

NL2SQL 模型	RPT	S-NUM	S-COL	AGG	W-NUM	W-COL	W-COL-OP	COND-OP
Seq2Seq	-	-	27.6%	39.4%	-	15.5%	36.2%	42.0%
Seq2SQL	-	-	73.4%	74.7%	-	35.5%	57.7%	71.5%
SQLNet	-	98.8%	88.0%	93.6%	91.5%	60.6%	85.5%	90.7%
TypeSQL	-	96.7%	91.0%	92.0%	91.7%	65.7%	87.1%	91.4%
SQLNet+BERT	-	97.1%	95.6%	96.0%	77.4%	58.1%	75.0%	80.3%
SQLNet+ Chinese-BERT-wwm	-	97.9%	95.9%	97.1%	90.2%	78.7%	85.1%	89.8%
SQLova	-	-	95.3%	90.1%	-	94.5%	96.1%	95.3%
X-SQL	-	-	96.1%	91.5%	-	96.7%	96.5%	96.2%
MyModel1	99.7%	99.3%	98.0%	98.1%	97.6%	97.1%	97.0%	97.2%

*注：-表示模型中不包含该子任务。

经过表 4.3 的对比实验数据可以看出，本文的列值抽取模型在所有子任务中都具有更好的表现，同时也都能达到不错的准确率。下面对各个模型在不同子任务上的表现详细分析说明。

实验结果表明，TypeSQL 模型与 SQLNet 模型相比，在 8 个子任务中的表现都有提升，其中 S-COL 和 W-COL 两个子任务上提升比较明显，主要是因为 TypeSQL 在 SQLNet 的基础上对自然语言问句文本中的实体和数字类型信息进行了建模，该实验结果有力证明了含有外部实体类型信息可以帮助模型提升列名识别能力。

表 4.3 中 SQLNet 模型的实验结果与 SQLNet+BERT 模型的实验结果相比，数字相关的两个子任务（S-NUM 和 W-NUM）在准确率上有所降低，并且在 W-NUM 子任务上较大的差距直接影响了 WHERE 语句列的选择，所以 W-NUM 准确率较低时

WHERE 语句的其他子任务准确率也会降低。而在 SELECT 语句选择列方面，SQLNet+BERT 模型在 S-NUM 降低的情况下仍然比 SQLNet 模型表现更好。该实验充分说明 BERT 模型在数字相关的标签子任务上未能发挥作用，而在文本类标签上则有所改善。

表 4.3 中 SQLNet+ Chinese-BERT-wwm 模型对比 SQLNet+BERT 模型的实验结果，两组实验在 S-NUM、S-COL 以及 AGG 等子任务上的准确率都很接近，而 SQLNet+ Chinese-BERT-wwm 模型在其他与 WHERE 语句相关的子任务（比如 W-COL、W-NUM）上准确率有明显的提升。由于 Chinese-BERT-wwm 模型在样本生成策略上有所改进，并且在实验中对各子任务具有积极影响，所以本文使用 SQLNet+ Chinese-BERT-wwm 模型作为预训练模型。

表 4.3 中 SQLova 和 X-SQL 模型与前面几个模型在各子任务上提升效果都比较明显，有力的证明了对数据库表中的列名和自然语言问句信息加以利用更能提升 NL2SQL 的模型效果。而 X-SQL 与 SQLova 相比，使用语义增强模式编码器将结果提升到新的高度。

从表 4.3 中可以看出，本文提出的模型在各个子任务上的表现比对比模型都要更好，首先是因为 Chinese-BERT-wwm 模型在中文文本特征提取方面具有更大的优势，子任务之间的依赖性也使得某个任务准确率非常高时，其他子任务也会因此有所提升；其次是本文在模型中将数据库表信息作为模型编码的输入，使得模型能够更精确地输出各任务的结果；另外的原因是本文模型中加入了其他模型没有的 RPT 子任务，借助 RPT 的预测值可以辅助模型正确的选中 SQL 语句应该选择的列，列名是否被重复使用也能在一定程度提高 WHERE 语句中列的选择准确率。

在实验过程中发现 RPT、S-NUM、S-COL 和 AGG 等子任务在模型开始训练时便有较高的准确率，而 W-NUM、W-COL、W-COL-OP 和 COND-OP 等子任务的准确率在训练初期较低，并且需要更长的训练时间才能达到较高准确率。此种情况直接说明与 SELECT 语句相关的子任务比 WHERE 语句相关的子任务的预测更为简便。所以，WHERE 语句相关的子任务在更大程度上影响了本文模型在性能上的提升，而 WHERE 语句的子任务受到列名的影响最大，在有重复列名使用的情况下，模型很难

预测选择数据库表中的列名。

通过对数据集中验证集的分析发现，有 540 条验证集数据均在自然语言问句中重复使用了列名。所以在此种数据集下，本文新增了下述预测 WHERE 语句列名的对比实验。该组实验将 SQLNet+BERT 模型、SQLNet+Chinese-BERT-wwm 模型和本文提出的列值抽取模型（MyModel1）共同作为对比模型，且实验数据均为列名复用的数据。实验结果如表 4.4 所示，说明本文模型在列名使用情况复杂的条件下，对列名的预测仍然领先于其他模型。

表 4.4　列名复用下预测结果比较实验

NL2SQL 模型	WHERE 语句列名预测准确率
SQLNet+BERT	89.9%
SQLNet+Chinese-BERT-wwm	93.1%
MyModel1	95.8%

4.4.3　条件值抽取模型训练

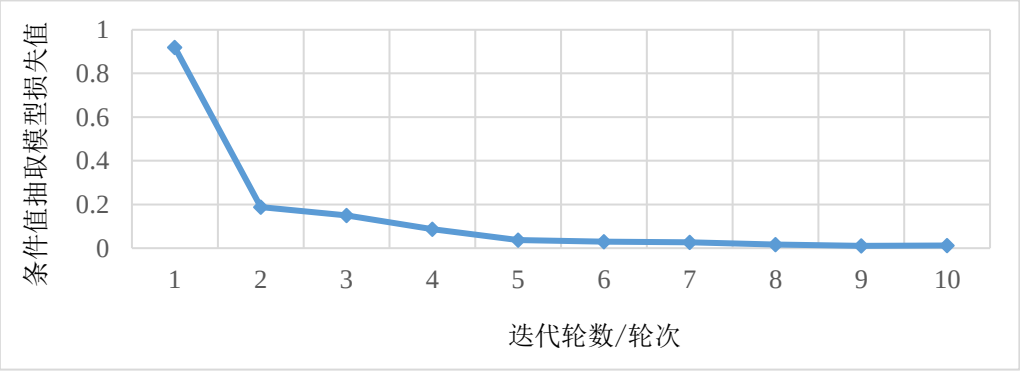

图 4.4　条件值抽取模型训练过程中损失值变化图

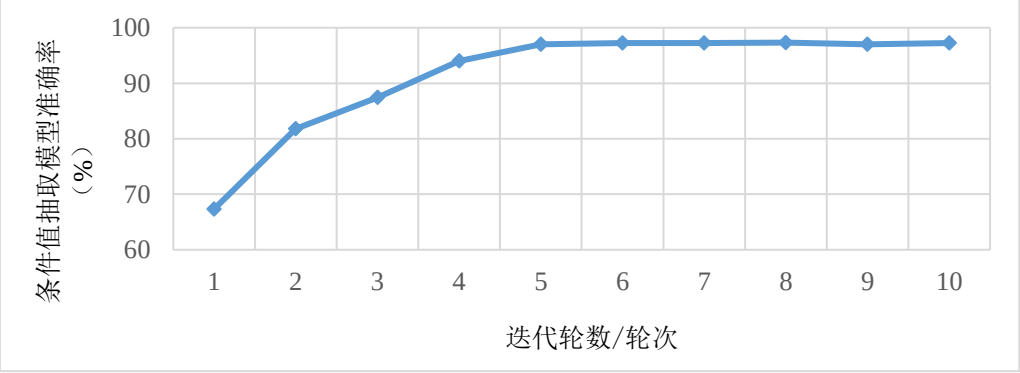

图 4.5　条件值抽取模型训练过程中准确率变化图

在列值抽取模型训练完成后，才可以进行条件值抽取模型的训练。本文在条件值抽取模型训练过程中，模型的损失值和准确率随迭代轮数 epoch 变化分别如图 4.4 和图 4.5 所示，同样可以看出列值抽取模型在训练过程中损失值的下降趋势以及准确率的上升趋势，并且最终达到趋于平稳的状态。

4.4.4 条件值抽取模型结果分析

本文使用的对比模型中，Seq2Seq 模型中对条件值抽取任务仍然使用传统的方法，即采用序列生成任务的方式来完成。Pointer Networks 被提出后，之后的 Seq2SQL、SQLNet 和 TypeSQL 模型在生成条件值时，都是基于 Pointer Networks 来完成的。

模型生成 SQL 语句的条件值通常使用以下两种方式，分别是序列生成方式和 Pointer Networks 方式。本文在条件值抽取模型中采用的是标签分类的方法，并将条件值抽取模型与其他模型基于验证集进行对比实验，实验结果如表 4.5 所示。

表 4.5 条件值抽取模型在验证集上的准确率

NL2SQL 模型	条件值生成准确率
Seq2Seq	18.52%
Seq2SQL	22.89%
SQLNet	46.95%
TypeSQL	62.27%
SQLNet+BERT	77.23%
SQLNet+Chinese-BERT-wwm	78.11%
SQLova	95.70%
X-SQL	96.30%
MyModel2	97.32%

从表 4.5 所示的试验数据中可以看出，Seq2Seq 算法的条件值抽取效率不高，此种情况是由于该算法基于序列生成，而序列生成问题具有极大复杂性，其他模型效果提高的原因是模型把问题分解为序列标记和分类任务。由于条件值属于 SQL 语句中 WHERE 语句的内容，而在表 4.4 的实验数据中发现，Seq2SQL 模型在 W-COL 子任务上预测准确率比 SQLNet 模型和 TypeSQL 模型的准确率低很多，所以在条件值抽取实验中，Seq2SQL 模型的准确性也比另外两个模型准确性要低。通过对比

SQLNet+BERT 模型和 SQLNet+Chinese-BERT-wwm 模型的实验结果，发现后者的准确率更高，所以在 NL2SQL 任务中，Chinese-BERT-wwm 模型对中文中语义特征的学习具有积极作用。SQLova 和 X-SQL 的实验结果表明，子任务之间的依赖性也使得列值抽取模型中 WHERE 语句子任务准确率非常高时，条件值的子任务准确率也会因此有所提升。

该表中的实验结果表明，本文提出的条件值抽取模型改善了传统序列生成方式和 Pointer Networks 方式的劣势，并且通过使用外部特征编码与预训练模型的输入融合的方式实现了最好的性能，因此本文决定使用此种方式结合分类模型作为条件值抽取的最终模型。

4.4.5 总体实验结果分析

本文实验最终合成的结果如表 4.6 所示。通过 SQL 语句的比较可以看到本文中模型预测生成的 SQL 语句与中文 NL2SQL 数据集中真实 SQL 语句完全一致。

表 4.6 本文模型生成的 SQL 语句与真实 SQL 语句对比

自然语言问句	本文模型预测生成的 SQL	真实 SQL
搜房网和人人网的周涨跌幅是多少	SELECT 周涨跌幅 from TABLE WHERE 名称 == 搜房网 or 名称 == 人人网	SELECT 周涨跌幅 from TABLE WHERE 名称 == 搜房网 or 名称 == 人人网
中天城投和福星股份的 2012 每股收益是多少	SELECT EPS2012E from TABLE WHERE 公司 == 中天城投 or 公司 == 福星股份	SELECT EPS2012E from TABLE WHERE 公司 == 中天城投 or 公司 == 福星股份

为了比较合理地分析 NL2SQL 模型生成的 SQL 语句的准确率，本文采用了公式(4.4)中的逻辑准确率（Acc_{LA}）、公式(4.6)中的执行准确率（Acc_{EA}）和公式(4.7)中的平均准确率（Acc_{MA}）作为标准来评估 NL2SQL 模型的效果。本文实验过程中分别将对比模型和本文模型生成的 SQL 语句与中文 NL2SQL 数据集中真实 SQL 语句进行比较，在上述评估标准上的实验结果如表 4.7 所示。

通过对表 4.7 中实验结果进行分析可以看出，每个模型的执行准确率（Acc_{EA}）都比逻辑准确率（Acc_{LA}）要高，说明执行准确率存在高估的可能，通过 NL2SQL 模型生成的 SQL 语句与数据集中标准 SQL 语句不同时也可能出现相同的执行结果；而逻辑准确率则存在低估的可能，此种情况是由于两个 SQL 语句中 SELECT 列的顺序

不同，其含义和执行结果也是相同的，通过 NL2SQL 模型生成的 SQL 语句与数据集中标准 SQL 语句就可能存在此种情况。所以本文引入了平均准确率（Acc_{MA}）来更合理的评估 NL2SQL 模型的效果。

表 4.7　总体实验结果

NL2SQL 模型	验证集			测试集		
	Acc_{LA}	Acc_{EA}	Acc_{MA}	Acc_{LA}	Acc_{EA}	Acc_{MA}
Seq2Seq	8.90%	10.31%	9.61%	8.11%	9.83%	8.97%
Seq2SQL	19.72%	23.65%	21.69%	19.24%	22.41%	20.83%
SQLNet	27.41%	30.14%	28.78%	26.97%	29.45%	28.21%
TypeSQL	32.33%	35.46%	33.90%	29.78%	31.11%	30.45%
SQLNet+BERT	51.92%	60.83%	56.38%	50.27%	57.33%	53.80%
SQLNet+ Chinese-BERT-wwm	52.77%	63.12%	57.95%	50.33%	59.47%	54.90%
SQLova	80.30%	83.20%	81.75%	81.30%	82.60%	81.95%
X-SQL	82.50%	86.70%	84.60%	81.80%	86.40%	84.10%
MyModel	83.39%	87.90%	85.65%	82.62%	87.92%	85.27%

4.5　本章小结

本章首先介绍本文实验所使用数据集和数据预处理的方法，说明本文实验在训练模型时使用的参数和实验环境。其次说明本文对实验结果的评估标准，在已有的自然语言转 SQL 语句工作中，本章选择了具有代表意义的工作来作为对比模型。由于本文算法将自然语言转 SQL 语句的工作分为两个模型，实验部分对该两个模型分别进行训练，通过完成训练的模型与对比模型在各个子任务的实验数据说明，本文所使用的 Chinese-BERT-wwm 模型能够有效提升中文 NL2SQL 模型性能。本章最后对总体实验结果进行对比分析，总结得出，本文所提方法对 SQL 语句各部分子语句的预测结果均有积极作用，且在一定程度上提高了中文 NL2SQL 模型生成完整 SQL 语句的准确率。

5 总结和展望

5.1 工作总结

数据库系统可以有效组织和管理数据，为便于通过自然语言与数据库系统人机交互，自然语言转换成结构化查询语句（NL2SQL）技术在业界和学术界引起极大重视。本文以深度学习为基础，面向中文自然语言生成 SQL 语句技术展开研究工作，对已有 NL2SQL 工作进行改进，主要工作总结如下。

（1）介绍了自然语言生成结构化查询语句（NL2SQL）任务的研究背景及意义，其有助于简单便捷的操作数据库和自然语言处理领域的发展。接着对 NL2SQL 国内外研究现状进行分析，深度学习方法为 NL2SQL 工作提供了良好的前提条件。通过分析 NL2SQL 方法，发现大多数 NL2SQL 工作是面向英文，中文 NL2SQL 工作较少，且现有的 NL2SQL 模型未能完全解决数据库中元数据信息识别问题和自然语言问句表述多样性问题。

（2）本文研究工作是面向中文的自然语言生成结构化查询语句（NL2SQL）问题，所以本文结合目前存在的实际落地应用对中文 NL2SQL 行业进行分析，包括目前存在的实际落地应用。此外，本文还通过对当前优秀的 NL2SQL 工作和强大的预训练模型进行详细介绍，分析如何利用现有的工作成果完成中文 NL2SQL 工作。对目前的 NL2SQL 问题相关概念进行归纳和总结，并概括了本文完成中文 NL2SQL 任务的方法流程。

（3）详细描述中文自然语言转结构化查询语句（NL2SQL）方法和模型结构，本文针对中文 NL2SQL 任务提出使用结合预训练语言模型和深度学习分类模型的方法，为中文 NL2SQL 任务构建两个深度学习网络模型，分别是列值抽取模型和条件值抽取模型。本文的列值提取模型可以获得 SQL 语句中除条件值以外其他所有内容，将数据库表结构信息融入到该模型中，并使用了更有约束力的子任务提高列值抽取模型对列名的预测能力，解决了数据库元数据信息识别问题；条件值抽取模型则引入数据库表内容信息和列值抽取模型的结果对 SQL 语句中条件值进行约束，解决了自然语言问句和数据库信息表述不一致的问题。

（4）通过在真实大规模 NL2SQL 数据集上进行对比实验，并进行了实验分析。实验结果证明，本文所提的模型比其他模型具有更高的准确率。由此证明本文方法在中文 NL2SQL 任务上具有较好效果。

5.2 工作展望

NL2SQL 任务具有重大研究意义和应用价值，本文对未来 NL2SQL 技术发展所做展望如下。

（1）本文通过使用大规模人工标注数据集完成中文 NL2SQL 模型训练，但在策划大规模数据集时往往需要消耗大量人力及时间成本，创建 NL2SQL 相关数据集更需要数据库专业知识。在 NL2SQL 实际应用场景中，往往只有少量 NL2SQL 数据，甚至没有 NL2SQL 数据可用，如何利用该数据完成 NL2SQL 模型训练是未来重要的研究方向。

（2）本文在处理中文 NL2SQL 任务时，生成 SQL 语句的过程本质上使用的还是槽值填充思想。但基于此种思想实现的方法大多只能按照填充模板顺序生成 SQL 语句，该方法对于多变的 SQL 语句来说并不稳定，而且模型训练过程繁琐。虽然语法分析是在深度学习之前尝试使用的方法，但如果结合深度学习方法和语法分析树，在 NL2SQL 任务上也能取得良好效果。

（3）随着自然语言处理技术不断发展，以及预训练语言模型的大规模使用，使得 NL2SQL 技术能够在实际场景中实现。但是在实际应用中仍然存在着许多问题，例如在自然语言的建模中更多的是使用联结式的深度学习等方式，而 SQL 语句则是形式化语言，人工智能技术尚未能完全弥合符号主义和联结主义之间的鸿沟。随着人工智能技术不断进步，各种方法层出不穷，NL2SQL 等语义分析问题在将来会被完美地解决，从而实现通过自然语言的交互方式实现与数据库的人机交互。

参考文献

[1] Jim Melton. Database Language sql. In: Handbook on Architectures of Information Systems, Springer, 2006: 105-132

[2] 罗枭. 基于深度学习的自然语言处理研究综述. 智能计算机与应用, 2020, 10(4): 133-137

[3] Karam Ahkouk, Mustapha Machkour. Towards an Interface for Translating Natural Language Questions to SQL: A Conceptual Framework from a Systematic Review. System, 2020, 12(4): 264-275

[4] 杨鹤标, 陈力. 自然语言向 SQL 代码的转化方法. 计算机工程, 2011, 37(23): 72-74

[5] 上官明霞, 朱珊珊, 陈晓亮, 王晶华, 郭光. 基于融合自然语言处理的语义分析方法研究. 计算机与网络, 2018, 44(20): 65-67

[6] Qingkai Min, Yuefeng Shi, Yue Zhang. A Pilot Study for Chinese SQL Semantic Parsing. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019), Hong Kong, China, November 3-7, 2019, Association for Computational Linguistics, 2019: 3650-3656

[7] Yoav Goldberg. 基于深度学习的自然语言处理（第一版）. 车万翔译, 郭江译, 张伟男译, 刘铭译. 北京: 机械工业出版社, 2018

[8] Ana-Maria Popescu, Oren Etzioni, Henry A. Kautz. Towards a Theory of Natural Language Interfaces to Databases. In: Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI 2003), Miami, Florida, USA, January 12-15, 2003, ACM, 2003: 149-157

[9] Alessandra Giordani, Alessandro Moschitti. Translating Questions to SQL Queries with Generative Parsers Discriminatively Reranked. In: COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, December 8-15, 2012, Mumbai, India, Indian Institute of Technology Bombay, 2012: 401-410

[10] Woods W, Kaplan R M, Nash-Webber B L. The Lunar Science Natural Language

Information System: Final Report. 1972

[11] Santiago Ontan. SHRDLU: A Game Prototype Inspired by Winograd's Natural Language Understanding Work. In: Proceedings of the Fourteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2018), November 13-17, 2018, Edmonton, Canada, AAAI Press, 2018: 268-270

[12] Roger C. Schank, Neil M. Goldman, Charles J. Rieger III, Christopher Riesbeck. Margie: Memory Analysis Response Generation, and Inference on English. In: Proceedings of the 3rd International Joint Conference on Artificial Intelligence. Standford, California, USA, August 20-23, 1973, William Kaufmann, 1973: 255-261

[13] Gary G. Hendrix, Earl D. Sacerdoti, Daniel Sagalowicz, Jonathan Slocum. Developing a Natural Language Interface to Complex Data, ACM Translate Database System. 1978, 3(2): 105-147

[14] Barbara J. Grosz. Team: A Transportable Natural-Language Interface System. In: 1st Applied Natural Language Processing Conference (ANLP 1983), Miramar-Sheraton Hotel, Santa Monica, California, USA, February 1-3, 1983, ACL, 1983: 39-45

[15] Katrin Affolter, Kurt Stockinger, Abraham Bernstein. A Comparative Survey of Recent Natural Language Interfaces for Databases. VLDB JOURNAL, 2019, 28(5): 793-819

[16] Lukas Blunschi, Claudio Jossen, Donald Kossmann, Magdalini Mori, Kurt Stockinger. SODA: Generating SQL for Business Users. VLDB Endowment, 2012, 5(10): 932-943

[17] Saeedeh Shekarpour, Edgard Marx, Axel C. Ngomo, Soren Auer. SINA: Semantic Interpretation of User Queries for Question Answering on Interlinked Data. Journal of Web Semantics, 2015, 30: 39-51

[18] Hannah Bast, Elmar Haussmann. More Accurate Question Answering on Freebase. In: Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM 2015), Melbourne, Victoria, Australia, October 19-23, 2015, ACM, 2015: 1431-1440

[19] Weiguo Zheng, Hong Cheng, Lei Zou, Jeffrey Xu Yu, Kangfei Zhao. Natural Language Question/Answering: Let Users Talk With the Knowledge Graph. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge

Management (CIKM 2017), Singapore, November 06-10, 2017, ACM, 2017: 217-226

[20] Esther Kaufmann, Abraham Bernstein. Evaluating the Usability of Natural Language Query Languages and Interfaces to Semantic Web Knowledge Bases. Journal of Web Semantics, 2010, 8(4): 377-393

[21] Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R. Mittal, Fatma "Ozcan. Athena: An Ontology-Driven System for Natural Language Querying Over Relational Data Stores. VLDB Endowment, 2016, 9(12): 1209-1220

[22] Sebastian Walter, Christina Unger, Philipp Cimiano, Daniel B"ar. Evaluation of a Layered Approach to Question Answering Over Linked Data. In: The Semantic Web - (ISWC 2012) - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part II, Springer, 2012: 362-374

[23] Hasan M. Jamil. Knowledge Rich Natural Language Queries Over Structured Biological Databases. In: Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics (BCB 2017), Boston, Massachusetts, USA, August 20-23, 2017, ACM, 2017: 352-361

[24] Mohnish Dubey, Sourish Dasgupta, Ankit Sharma, Konrad H"offner, Jens Lehmann. Asknow: a Framework for Natural Language Query Formalization in Sparql. In: The Semantic Web. Latest Advances and New Domains - 13th International Conference (ESWC 2016), Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings, Springer, 2016: 300-316

[25] S'ebastien Ferr'e. Sparklis: An Expressive Query Builder for Sparql Endpoints with Guidance in Natural Language. Semantic Web, 2017, 8(3): 405-418

[26] Anca Marginean. Question Answering over Biomedical Linked Data with Grammatical Framework. Semantic Web, 2017, 8(4): 565-580

[27] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), October 25-29, 2014, Doha, Qatar, 2014: 1724-1734

[28] Ilya Sutskever, Oriol Vinyals, Quoc V. Le. Sequence to Sequence Learning with

Neural Networks. In: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, 2014: 3104-3112

[29] Oriol Vinyals, Meire Fortunato, Navdeep Jaitly. Pointer Networks. In: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, 2015: 2692-2700

[30] Li Dong, Mirella Lapata. Language to logical form with neural attention. CoRR, 2016, abs/1601.01280

[31] Weicheng Ma, Kai Cao, Zhaoheng Ni, Peter Chin, Xiang Li. Sound Signal Processing with Seq2tree Network. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, May 7-12, 2018, European Language Resources Association (ELRA), 2018: 3132-3136

[32] Victor Zhong, Caiming Xiong, Richard Socher. Seq2sql: Generating Structured Queries from Natural Language using Reinforcement Learning. CoRR, 2017, abs/1709.00103

[33] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, et al. Rainbow: Combining Improvements in Deep Reinforcement Learning. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018), New Orleans, Louisiana, USA, February 2-7, 2018, AAAI Press, 2018: 3215-3222

[34] Semih Yavuz, Izzeddin Gur, Yu Su, Xifeng Yan. What It Takes to Achieve 100 Percent Condition Accuracy on WikiSQL. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31-November 4, 2018, Association for Computational Linguistics, 2018: 1702-1711

[35] Xiaojun Xu, Chang Liu, Dawn Song. SQLNet: Generating Structured Queries from Natural Language without Reinforcement Learning. CoRR, 2017, abs/1711.04436

[36] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, Dragomir R. Radev. Typesql: Knowledge-based Type-aware Neural Text-to-SQL Generation. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana,

USA, June 1-6, 2018, Short Papers, Association for Computational Linguistics, 2018: 588-594

[37] Li Dong, Mirella Lapata. Coarse-to-fine Decoding for Neural Semantic Parsing. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018), Melbourne, Australia, July 15-20, 2018, Long Papers, Association for Computational Linguistics, 2018: 731-742

[38] Shivaji Alaparthi, Manit Mishra. Bidirectional Encoder Representations from Transformers (BERT): a Sentiment Analysis Odyssey. CoRR, 2020, abs/2007.01127

[39] Michele Bevilacqua, Roberto Navigli. Quasi Bidirectional Encoder Representations from Transformers for Word Sense Disambiguation. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019), Varna, Bulgaria, September 2-4, 2019, INCOMA, 2019: 122-131

[40] Wonseok Hwang, Jinyeung Yim, Seunghyun Park, Minjoon Seo. A Comprehensive Exploration on WikiSQL with Table-aware Word Contextualization. CoRR, 2019, abs/1902.01069

[41] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, et al. Spider: A Large-scale Human-labeled Dataset for Complex and Cross-domain Semantic Parsing and Text-to-sql Task. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31-November 4, 2018, Association for Computational Linguistics, 2018: 3911-3921

[42] Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, et al. SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-domain Text-to-sql Task. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018, Association for Computational Linguistics, 2018: 1653-1663

[43] Dongjun Lee. Clause-wise and Recursive Decoding for Complex and Cross-domain Text-to-sql Generation. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019), Hong Kong, China, November 3-7, 2019, Association for Computational Linguistics, 2019: 6044-6050

[44] Ben Bogin, Jonathan Berant, Matt Gardner. Representing Schema Structure with

Graph Neural Networks for Text-to-sql Parsing. In: Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL 2019), Florence, Italy, July 28 - August 2, 2019, Long Papers, Association for Computational Linguistics, 2019: 4560-4565

[45] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. IEEE Trans, Neural Networks Learn, System, 2021, 32(1): 4-24

[46] Yujia Li, Daniel Tarlow, Marc Brockschmidt, Richard S. Zemel. Gated Graph Sequence Neural Networks. CoRR, 2016, abs/1511.05493

[47] Jun Xu, Haifeng Wang, Zhengyu Niu, Hua Wu, Wanxiang Che. A Key-phrase Aware End2End Neural Response Generation Model. In: Natural Language Processing and Chinese Computing - 8th CCF International Conference (NLPCC 2019), Dunhuang, China, October 9-14, 2019, Proceedings, Part II, Springer, 2019: 65-77

[48] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, et al. Towards Complex Text-to-sql in Cross-domain Database with Intermediate Representation. In: Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL 2019), Florence, Italy, July 28 - August 2, 2019, Long Papers, Association for Computational Linguistics, 2019: 4524-4535

[49] Jeong-Oog Lee, Doo-Kwon Baik. Semql: a Semantic Query Language for Multidatabase Systems. In: Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management, Kansas City, Missouri, USA, November 2-6, 1999, ACM, 1999: 259-266

[50] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, Matthew Richardson. Rat-sql: Relation-aware Schema Encoding and Linking for Text-to-sql Parsers. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020), Online, July 5-10, 2020, Association for Computational Linguistics, 2020: 7567-7578

[51] Richard Shin. Encoding Database Schemas with Relation-aware Self-attention for Text-to-sql Parsers. CoRR, 2019, abs/1906.11790

[52] Peter Shaw, Jakob Uszkoreit, Ashish Vaswani. Self-attention with Relative Position Representations. In: Proceedings of the 2018 Conference of the North American

Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018), New Orleans, Louisiana, USA, June 1-6, Short Papers, Association for Computational Linguistics, 2018: 464-468

[53] 刘译璟, 徐杰, 代其锋. 基于自然语言处理和深度学习的 NL2SQL 技术及其在 BI 增强分析中的应用. 中国信息化, 2019(11): 62-67

[54] 张琰. 一种基于 BERT 的中文 NL2SQL 模型[硕士学位论文]. 济南: 山东大学, 2020

[55] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. CoRR, 2018, abs/1810.04805

[56] Pengcheng He, Yi Mao, Kaushik Chakrabarti, Weizhu Chen. X-SQL: Reinforce Schema Representation with Context. CoRR, 2019, abs/1908.08113

[57] Xiaodong Liu, Pengcheng He, Weizhu Chen, Jianfeng Gao. Multi-task Deep Neural Networks for Natural Language Understanding. In: Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Long Papers, Association for Computational Linguistics, 2019: 4487-4496

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, et al. Attention is All You Need. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, California, USA, 2017: 5998-6008

[59] Guy Kushilevitz, Shaul Markovitch, Yoav Goldberg. A Two-stage Masked lm Method for Term Set Expansion. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020), Online, July 5-10, 2020, Association for Computational Linguistics, 2020: 6829-6835

[60] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, et al. Pre-training with Whole Word Masking for Chinese Bert. CoRR, 2019, abs/1906.08101