文章编号：2095-2163(2021)06-0109-08 中图分类号：TP391.2 文献标志码：A

# 基于混合排序填充网络的文本到指令序列翻译

张晓芳[1]，欧 睿[1]，曾钰城[2]，饶攀军[1]，陈 科[1]，郑 元[1]，张 雷[1]，李明艳[1]
（1 太极计算机股份有限公司，北京 100102；2 哈尔滨工业大学，哈尔滨 150001）

**摘 要**：本论文主要研究利用语言预训练模型和深度学习，来进行文本序列到指令序列的翻译。在文本序列到指令序列翻译的领域内，优势模型按结构划分主要分为 2 类，一种是端到端模型，另外一种是在编码器和解码器中间引入中间逻辑表示层。与以往的方法不同，本文提出了名为混合排序填充网络的新模型。该模型将问题与一列单独组合，利用语言预训练模型获取编码，然后根据选择的模板，划分并填充不同的子任务，生成完整的指令序列。这种方法很好地利用了语言预训练模型的特点，避免引入专门的中间语义表示层或者解码器，减少了模型复杂度。在自构建的文本到指令序列数据集上取得了很好的翻译效果，翻译结果的逻辑准确率可达到 89.1%。

**关键词**：文本序列到指令序列；语言预训练模型；混合排序填充网络；子任务计划

## Hybrid ranking filling network for text-to-SQL

ZHANG Xiaofang[1], OU Rui[1], ZENG Yucheng[2], RAO Panjun[1], CHEN Ke[1], ZHENG Yuan[1], ZHANG Lei[1], LI Mingyan[1]
(1 Taiji Computer Corporation Limited, Beijing 100102, China; 2 Harbin Institute of Technology, Harbin 150001, China)

**【Abstract】** In this paper, we study how to leverage pre-trained language models and deep learning network in Text-to-SQL. The previous approach can be divided into two categories according to the structure, Seq-to-Seq model and intermediate representation model. We merge the previous approach and propose a new approach called Hybrid Ranking Filling Network. In this approach, first, the encoder is given a NL question and one individual column. Second, this network breaks down the problem into different sub-tasks and fills them. Finally, this network assembles the sub-tasks outputs into a SQL query. Our approach avoids any ad-hoc additional encoding layers which are necessary in prior approaches, hence our approach is more elegant. Experiments on the dataset constructed by the author show that the proposed approach is very effective and the logical form accuracy can reach 89.1%.

**【Key words】** Text-to-SQL; pre-trained language models; Hybrid Ranking Filling Network; sub-tasks

## 0 引 言

在传统的以业务流为基础的软件中，用户会根据业务类型，调用业务对应的程序接口，完成业务需求。但随着业务需求的多样化，业务对应的程序接口也会变得愈加繁杂，此时用户为完成一项业务所需的操作步骤就会增多，导致工作效率下降和增加出错概率。针对于此，本文研究的指令序列转换，是把用户用自然语言表述的业务需求转换为可供系统执行的指令，通过执行指令完成用户所表达的业务需求，可以有效缩短操作步骤，提高工作效率和降低出错概率。

通常情况下，业务需求基本上都可以映射为对数据库的增、删、改、查。由于指令与 SQL 语句类似，则可将其转换成类似于 SQL 语句的表达方式，进而用 Text2SQL 和 NL2SQL 相关技术完成自然语言到指令的转换。

自然语言序列到指令序列的转换任务，在自然语言处理领域一直受到研究人员的关注，通常被称为 NLIDBs(Natural Language Interface to Databases)[1-2]。早期提出的系统大多都是基于人工编写的特定转换规则，只能应用到特定的数据库中[3-5]。后来的工作着重于构建一个能够泛化的转换系统，该系统可以用最少的人力即可用于多个数据库[6-8]。近年来，随着语义解析技术的进步，以端到端[9-10]模型为代表的方法，以及在此基础之上构建的高级神经网络的方法[11-12]，在大规模跨领域 Text-to-SQL 数据集上。例如，WikiSQL [13]和 Spider[14]，取得了非常好的效果。

针对上述问题，结合近年来有关 Text2SQL 的工作，本文对以往模型进行改进，提出了混合排序填充网络模型(Hybrid Ranking Filling Network, H-Net)。混合排序填充网络模型建模过程如下：

(1)利用语言预训练模型 BERT[15]对一个列与

作者简介：张晓芳(1973-)，女，硕士，高级工程师，主要研究方向：企业管理创新、人工智能、知识图谱；欧 睿(1984-)，男，硕士，工程师，主要研究方向：人工智能、自然语言处理、知识图谱等；曾钰城(1998-)，男，本科生，主要研究方向：人工智能、自然语言处理。

收稿日期：2021-05-10

问句进行联合编码,用以获取列与问句之间的关系;

(2)根据选择的 SQL 模板,将生成完整的 SQL 语句任务,划分为若干个生成子 SQL 语句任务;

(3)计算列与问句在不同子 SQL 语句中的相似度,并对计算的结果进行排序;

(4)针对不同的子 SQL 语句的生成任务,利用排序结果和其对应的解码方式进行解码,生成子 SQL 语句。

(5)利用子 SQL 语句填充 SQL 模板,生成完整的 SQL 语句。

混合排序填充模型与传统端到端模型的不同在于,混合排序填充模型不需要额外的编码器和解码器,也不需要引入额外的中间表示层,这些特点降低了模型复杂度,提高了模型泛化能力。为了验证本文提出方法的有效性,构建了文本序列到指令序列的数据集,在这一数据集上的实验结果,验证了本文方法的有效性。

## 1 相关工作

近年来,有关 Text2SQL 的工作主要可以分为端到端翻译、中间语义表示、SQL 子任务预测、中间语义表示等方法。下面分别介绍这 4 类模型,并且评价其优缺点,最后比较 H-Net 与这 4 类模型的联系和区别。

### 1.1 端到端翻译模型

端到端翻译模型主要思想是,通过端到端模型生成具有 SQL 语法结构的草图,然后利用问句中的内容填充该草图[13-16]。方法可以简单概括为以下步骤:

(1)将输入问句和数据库表分别进行编码,计算问句对数据库表的注意力,生成注意力表示作为模型输入,然后通过解码器生成一张粗略的草图;

(2)通过对齐问句与草图,填充草图中丢失的细节。端对端模型对语句顺序敏感,但 SQL 语句对顺序不敏感。例如,条件语句中交换 2 个条件顺序,对 SQL 语句并没有影响,但却会影响端对端模型的生成过程。

### 1.2 中间语义表示模型

中间语义表示模型主要思想是将问句和所有列拼接,进行编码获取输入,然后通过基于语法(Grammar Based)的解码器,生成树状结构的中间语义表示,最后基于中间表示再进行一次解码,生成完整的 SQL 语句[11-12,17]。方法可以简单概括为以下步骤:

(1)将问句与数据库进行链接(Schema Linking),找出问句中的与数据库对应的表、列和值序列,然后将数据库表中所有列与问句各自进行编码并且拼接作为输入;

(2)利用基于语法的解码器,通过 ApplyRule,GenToken 2 类规则,生成树状结构的中间语义表示;

(3)对中间语义表示再进行一次解码,生成完整的 SQL。

中间语义表示方法需要专门的基于语法的解码器、引入专门的中间语义表示层,增加模型复杂度。此外,引入中间语义不能完全表示问句内容,会造成语义损失[18]。

### 1.3 SQL 子任务预测模型

SQL 子任务预测模型主要思想,是将 SQL 语句拆解成若干个子 SQL 语句,然后分别生成子 SQL 语句,最后组合成一个完整的 SQL 语句[19-20]。这类方法可以简单概括为以下步骤:

(1)将 SQL 语句拆分为 SELECT-column、SELECT-aggregation、WHERE - number、WHERE - column、WHERE-operator、WHERE-value 等子 SQL 语句;

(2)每个子 SQL 语句都对应着一个子任务,每个子任务相当于一个分类任务。子任务之间各自独立,各自生成对应的子 SQL 语句,最后组合成一条完整的 SQL 语句。

SQL 子任务预测模型的子任务划分,是通过人工定义规则完成的,无法拓展到复杂的 SQL 语句;SQL 子任务预测模型没有充分利用句与列之间的依赖关系,没有充分利用任务之间的依赖关系。

### 1.4 语言预训练模型

模型的主要思想,是用语言预练习模型 BERT[15]捕捉问句与列之间的关系,利用问句与列之间的关系生成 SQL 语句[21-23]。方法可以简单概括为以下步骤:

(1)将问句和所有列拼接[21-22],或者将问题与单个列进行拼接[23]作为输入,利用语言预训练模型获取编码得到输出,获取问句和列之间的关系;

(2)用类似 SQL 子任务预测的方法,将完整 SQL 分成若干个子 SQL 语句;

(3)对于每一个子 SQL 语句,通过对语言预处理模型的输出,用线性神经网络[23]或 LSTM(Long Short-Term Memory)[24]进行解码,填充子 SQL 语句,最后合并子 SQL 语句,得到完整的子 SQL 语句。

这类方法的缺点是:SQL 子任务的划分是通过

人工定义规则完成；无法拓展到复杂的 SQL 语句；需要额外的 LSTM 解码器完成 SQL 语句生成。

### 1.5　H-Net 模型与上述 4 类方法的联系与区别

本文提出混合排序填充网络模型（H-Net）的方法与文献[19-20,23]方法类似，采用了语言预训练模型和 SQL 语句子任务划分。但主要有以下不同：

（1）引入模板生成层，可以根据训练数据中的 SQL 语句，抽取出 SQL 模板，让模型可以根据输入问句不同，选择不同的模板，进而依据模板可以划分出不同的 SQL 子任务，在此基础上生成结构复杂的 SQL 语句；

（2）拓展了填充 SQL 子语句的方法，可以生成复杂的 SQL 语句。

## 2　混合排序填充网络模型

在本节中，将详细介绍在混合排序填充模型如何进行模板抽取、模板选择、子任务划分和每类子任务的完成方式，最后介绍了如何进行模型训练和 SQL 语句预测。

### 2.1　模板抽取

首先，为 SQL 语句定义上下文无关文法，如图 1 所示。

root →s rel s
s →SELECT scol FROM table
WHERE wcol ORDER BY ocol s| φ
scol →agg col, scol|φ
wcol →col op value, wcol |col link value, wcol|
col link s, wcol|φ
ocol →col sc,ocol|φ
table →T, table|φ
col→C,col| φ
value→V, value|φ
agg→MAX|MIN|COUNT|AVG|SUM|φ
op→=|>=|...|LIKE|BETWEEN
sc→ ASC|DESC|φ
link→IN|NOT IN|EXIST|NOT EXIST
rel→UNION|EXCEPT|INTERSECT|φ

**图 1　SQL 语句的上下文无关文法**

**Fig. 1　Context-free grammar for SQL**

与 Grappa[25] 类似，对于训练集中 SQL 语句 $s$，用上下文无关文法对其进行解析。对于每一条 SQL 语句 $s$，都会得到一个 SQL 语句的语法结构。

例如，对于 SQL 语句：

select class_type from cource where id>=100 order by class_type asc
union
select class_type from cource_opened where id>=100 order id desc

利用上下文无关文法进行解析后，得到如图 2 所示的 SQL 语句的语法结构，并依据该语法结构进行聚类。对于聚类结果中的每一类，取该类覆盖率最高的 SQL 语句的语法结构，作为一个模板 Template，将所有模板记为一个模板集合 $T$。

SQL:{
SELECT:[($agg_{11}$,$scol_{11}$)]
FROM:[$table_{11}$]
WHERE:[($wcol_{11}$,$op_{11}$,$value_{11}$)]
ORDER:[($ocol_{11}$,$or_{11}$)]
SET:[rel]
SELECT:[($agg_{21}$,$scol_{21}$)]
FROM:[$table_{21}$]
WHERE:[($wcol_{21}$,$op_{21}$,$value_{21}$)]
ORDER:[($ocol_{21}$,$sc_{21}$)]
}

**图 2　SQL 语句的语法结构**

**Fig. 2　The syntax structure of an SQL**

### 2.2　输入表示

#### 2.2.1　定义

问句可以表示为：$q$

数据库中的一列可以表示为：

$$c_i = Concat(type_{c_i}, t_{c_i}, c_i, note^t_{c_i}, note^c_{c_i})$$

其中：$type$ 表示列的类型，$type \in \{string, number, date, ...\}$；$t$ 为表名；$c$ 为列名；$note^t_{c_i}$ 为注释；$note^c_{c_i}$ 为列注释；$Concat(\cdot)$ 表示连接操作。

将问句 $q$ 与列 $c_i$ 组合起来，得到输入对（$c_i$，$q$）。输入对经过分词后，得到一个词序列：

$$[CLS], x_1, x_2, ..., x_m, [SEP], y_1, y_2, ..., y_n, [SEP]$$

其中：$x_1, x_2, ..., x_m$ 表示列 $c_i$ 的词序列；$y_1, y_2, ..., y_n$ 表示问句 $q$ 的词序列；将 $[CLS], x_1, x_2, ..., x_m, [SEP], y_1, y_2, ..., y_n, [SEP]$ 作为模型输入，记为 $input$。

#### 2.2.2　编码

采用 BERT 语言预训练模型对输入进行编码，用于捕捉问句与列之间的关系，编码后将获得一个隐藏状态序列。

$$h_{[CLS]}, h^{c_i}_1, h^{c_i}_2, ..., h^{c_i}_m, h_{[SEP]}, h^q_1, h^q_2, ..., h^q_n, h_{[SEP]} = BERT(input). \quad (1)$$

### 2.3　模型

H-Net 模型由语言预训练模型 BERT、模板选择和模板填充 3 部分构成。模型结构如图 3 所示。

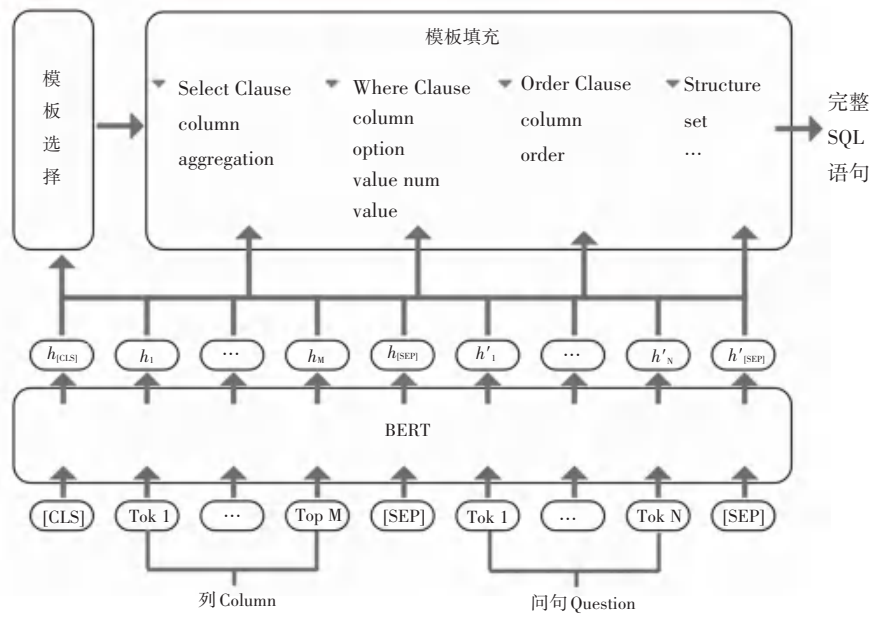


**图 3 H-Net 模型总体结构**

**Fig. 3 The structure of H-Net**

2.3.1 模板选择

模板选择可以认为是一种分类任务,对于给定问句 $q$, 其对应模板记为 $\tau$, 则有:

$$\tau = \arg\max_{\tau} P(\tau \mid q) = \arg\max_{\tau} \sum_{c_i} P(\tau \mid c_i, q) P(c_i \in R_c \mid q) \quad (2)$$

$$P(\tau = k \mid c_i, q) = \mathrm{softmax}(W^{template}[k,:] \cdot h_{[CLS]}) \quad (3)$$

$P(c_i \in R_c \mid q)$ 是问句 $q$ 与列 $c_i$ 在完整 SQL 语句中相似度。

2.3.2 子任务划分

假设一个 SQL 语句模板如图 4 所示。

```
SQL:{
SELECT:[(agg1,scol1),(agg2,scol2),...]
FROM:[table1,table2,...]
WHERE:[(wcol1,op1,value1),(wcol2,op2,value2),...]
ORDER:[(ocol1,or1),(ocol2,or2)]
SET:[rel]
...
}
```

**图 4 一个 SQL 语句模板样例**

**Fig. 4 A template of SQL**

根据 SQL 语句模板,可以将预测 SQL 语句拆分为 3 类子任务:

(1)列独立任务:预测选择子句所涉及的列 $S_c$, 条件子句中所涉及的列 $W_c$, 排序子句中所涉及的列 $O_c$ 和完整 SQL 语句中所涉及的列 $R_c$;

(2)列相关任务:预测函数运算符 *aggregation*、条件操作符 *operator*、条件对应的值 *value*、条件对应值的个数 *value_num* 等;

(3)结构相关任务:预测集合运算符 *rel*, 连接运算符 *link* 等。

2.3.3 列与问句相似度计算

对于问句 $q$, 记 $S_c$ 是在选择子句 *select_clause* 中所涉及的列; $W_c$ 是在条件子句 *where_clause* 中所涉及的列; $O_c$ 是在排序子句 *order_clause* 中所涉及的列; $R_c$ 是在完整的 SQL 语句中所涉及的列, $R_c = S_c \cup W_c \cup O_c$。

对于列 $c_i$ ,需要计算列 $c_i$ 与问句 $q$ 在不同 SQL 子句中的相似度,进而确定列 $c_i$ 在对应的 SQL 子句中是否出现。以下是 4 种不同相似度的计算方式:

在选择子句 *select_clause* 中,列 $c_i$ 与问句 $q$ 相似度记为 $P(c_i \in S_c \mid q)$:

$$P(c_i \in S_c \mid q) = \mathrm{sigmod}(\mathrm{W}_{sc} \cdot h_{[CLS]}). \quad (4)$$

在条件子句 *where_clause* 中,列 $c_i$ 与问句 $q$ 相似度记为 $P(c_i \in W_c \mid q)$:

$$P(c_i \in W_c \mid q) = \mathrm{sigmod}(\mathrm{W}_{wc} \cdot h_{[CLS]}). \quad (5)$$

在排序子句 *order_clause* 中,列 $c_i$ 与问句 $q$ 相似度记为 $P(c_i \in O_c \mid q)$:

$$P(c_i \in O_c \mid q) = \mathrm{sigmod}(\mathrm{W}_{oc} \cdot h_{[CLS]}). \quad (6)$$

在完整的 SQL 语句中,列 $c_i$ 与问句 $q$ 相似度记为 $P(c_i \in R_c \mid q)$:

$$P(c_i \in R_c \mid q) = \mathrm{sigmod}(\mathrm{W}_{rc} \cdot h_{[CLS]}). \quad (7)$$

2.3.4　列独立任务

列无关任务，是指在预测不同 SQL 子句中所涉及的列。如：$S_c$、$W_c$ 和 $O_c$。2 种预测方式为：一是设置一个阈值，对于每一类子句，只保留与问句相似度超过阈值的列；二是对于每一类子句，预测一个 $n$ 值，只保留与问句相似度排名前 $n$ 的列。对于第二种方法，$n$ 的计算过程如下：

(1)选择子句中的列数量 *select_num*，记为 $\hat{n}_s$：

$$\hat{n}_s = \arg\max_{n_s} P(n_s \mid q) = \arg\max_{n_s} \sum_{c_i} P(n_s \mid c_i, q) P(c_i \in S_c \mid q), \quad (8)$$

$$P(n_s = k \mid c_i, q) = \text{softmax}(W^{ns}[k,:] \cdot h_{[CLS]}). \quad (9)$$

其中，$P(n_s = k \mid c_i, q)$ 相当于一个分类任务。

(2)条件子句中的列数量 *where_num*，记为 $\hat{n}_w$：

$$\hat{n}_w = \arg\max_{n_w} P(n_w \mid q) = \arg\max_{n_w} \sum_{c_i} P(n_s \mid c_i, q) P(c_i \in W_c \mid q), \quad (10)$$

$$P(n_w = k \mid c_i, q) = \text{softmax}(W^{nw}[k,:] \cdot h_{[CLS]}). \quad (11)$$

其中，$P(n_w = k \mid c_i, q)$ 相当于一个分类任务。

(3)排序子句中的列数量 *order_num*，记为 $\hat{n}_o$：

$$\hat{n}_o = \arg\max_{n_o} P(n_o \mid q) = \arg\max_{n_o} \sum_{c_i} P(n_o \mid c_i, q) P(c_i \in O_c \mid q). \quad (12)$$

$$P(n_o = k \mid c_i, q) = \text{softmax}(W^{no}[k,:] \cdot h_{[CLS]}). \quad (13)$$

其中，$P(n_o = k \mid c_i, q)$ 相当于一个分类任务。

2.3.5　列相关任务

列相关任务，可以认为是一种分类任务，以下是 4 种列相关任务的计算方式：

(1)选择子句中函数运算符 $a_j$：

$$P(a_j \mid c_i, q) = \text{softmax}(\mathrm{W}^{agg}[j,:] \cdot h_{[CLS]}), \quad (14)$$

(2)条件子句中的条件操作符 $o_j$：

$$P(o_j \mid c_i, q) = \text{softmax}(\mathrm{W}^{op}[j,:] \cdot h_{[CLS]}). \quad (15)$$

(3)条件子句中的列对应的值 *value*：

首先，需要预测条件对应值的个数 *value_num* 记为 $n_v$，

$$P(n_v = j \mid c_i, q) = \text{softmax}(\mathrm{W}^{nv}[j,:] \cdot h_{[CLS]}). \quad (16)$$

对于条件对应的第 $k$ 个值 $value_k$，只需要指出 $value_k$ 在问句中开始位置和结束位置：

开始位置：

$$P(y_j = start \mid c_i, q) = \text{softmax}(\mathrm{W}_k^{start} \cdot h_j^q). \quad (17)$$

结束位置：

$$P(y_j = end \mid c_i, q) = \text{softmax}(\mathrm{W}_k^{end} \cdot h_j^q). \quad (18)$$

(4)排序子句中的运算符 $or_j$：

$$P(or_j \mid c_i, q) = \text{softmax}(\mathrm{W}^{order}[j,:] \cdot h_{[CLS]}). \quad (19)$$

2.3.6　结构相关任务

(1)对于集合运算符 *rel*：

$$\hat{rel} = \arg\max_{rel} P(rel \mid q) = \arg\max_{rel} \sum_{c_i} P(rel \mid c_i, q) P(c_i \in R_c \mid q), \quad (20)$$

$$P(rel = k \mid c_i, q) = \text{softmax}(W^{rel}[k,:] \cdot h_{[CLS]}). \quad (21)$$

(2)对于连接运算符 *link*：

$$\hat{link} = \arg\max_{link} P(link \mid q) = \arg\max_{link} \sum_{c_i} P(link \mid c_i, q) P(c_i \in R_c \mid q), \quad (22)$$

$$P(link = k \mid c_i, q) = \text{softmax}(W^{link}[k,:] \cdot h_{[CLS]}). \quad (23)$$

其中，$P(rel = k \mid c_i, q)$、$P(link = k \mid c_i, q)$ 均相当于一个分类任务。

### 2.4　训练与预测

2.4.1　训练阶段

输入 $(q_i, c_j)$。其中，$q_i$ 是第 $i$ 个问句，$U_C = \{c_1, c_2, \ldots, c_L\}$ 是数据库表中所有列，$c_j \in U_C$。

标签：与 $q_i$ 对应的 SQL 语句。

优化目标：最小化每一类子任务的交叉熵损失

$$\min \sum \text{CrossEntropy}(\text{sub\_task}). \quad (24)$$

2.4.2　预测阶段

(1)通过式(2)选择模板，进行 SQL 语句子任务划分；

(2)根据式(4)计算列与问句在选择子句中的相似度、排序；通过式(8)预测选择子句中的列的数量 $\hat{n}_s$，取相似度前 $\hat{n}_s$ 的列 $(cs_1, cs_2, \ldots, cs_{\hat{n}_s})$，根据式(14)计算每个列对应的函数运算符，得到选择子句 *select_clause*：

$$[(cs_1, agg_1), (cs_2, agg_2) \ldots, (cs_{\hat{n}_s}, agg_{\hat{n}_s})]$$

(3)根据式(5)计算列与问句在条件子句中的相似度、排序；通过式(10)预测条件子句中列的数量 $\hat{n}_w$，取相似度前 $\hat{n}_w$ 的列 $(cw_1, cw_2, \ldots, cw_{\hat{n}_w})$，根据式(15)计算每个列对应的条件操作符，根据式(16)~(18)计算每个列对应的值，得到条件子句 *where_clause*：

$$[(cw_1, op_1, (value_{11}, \ldots)), (cw_2, op_2, (value_{21}, \ldots)) \ldots, (cw_{\hat{n}_w}, op_{\hat{n}_w}, (value_{\hat{n}_w 1}, \ldots))]$$

(4)根据式(6)计算列与问句在排序子句中的相似度；通过式(12)预测排序子句中列的数量 $\hat{n}_o$，取相似度前 $\hat{n}_o$ 的列 $(co_1, co_2, \ldots, co_{\hat{n}_w})$，根据式(19)计算每个列对应的操作符，得到排序子句 *order_clause*：

$$[(co_1, or_1), (co_2, or_2) \ldots, (\hat{co_{n_o}}, \hat{or_{n_o}})]$$

(5)通过式(20)和(22)预测 SQL 语句的结构信息,得到 $\hat{rel}$ 和 $\hat{link}$;

(6)根据上述子句所选择出的列,可知这些列所对应的表。即 $S$ 是上述子句所对应的表, $S = [t_1, t_2, \ldots, t_{n_t}]$, 就可以得到 *from* 子句:

$$[t_1, t_2, \ldots, t_{n_t}]$$

(7)根据表 $S$ 之间的外键和主键的关系,找到表 $S$ 之间的最短路径,将最短路径上表之间的连接条件加入到条件子句中。

经过上述步骤,可以得到不同 SQL 的子句。根据 SQL 子句类型,将 SQL 子句填充到模板中对应的位置,最后得到完整的 SQL 语句。

### 2.5 执行指导与 SQL 语句剪枝

在 SQL 语句的预测阶段,部分生成的 SQL 语句也可以被执行。可根据部分生成的 SQL 语句,以及在数据库上执行的结果,调整 SQL 语句的生成,这被称为执行指导(Execution Guided, EG)[26]。执行指导的作用相当于剪枝,限制模型在解空间中的搜索范围,去除不满足要求的搜索路径。

## 3 指令序列转换

### 3.1 数据集说明

本文构建的文本到指令序列数据集为 JSON 格式,采用 JSON 格式是为了方便表示指令序列。数据集包含 2 312 条(问句与 JSON 语句对)。其中,1 923条数据是一个问句对应一条 JSON 语句,389 条数据是一个问题对应 2 条 JSON 语句。问句类型多为基于多条件查询匹配的答案检索。

JSON 语句由 method、url、params 3 部分构成。params 内包含多个参数(param),每个参数 param 又由 name、option、value 组成。参数 param 的 2 种类型如下:

(1)直接参数:value 值直接出现在问句中。如图 5 中参数 create_time 和 create_org;

(2)间接参数:value 值不直接出现在问句中。如图 5 中的参数 columns。

对于这 2 种参数,在指令转换中有各自的处理方式。

问句与 JSON 语句对示例如图 5 所示。

### 3.2 指令转换

为了使用 Text2SQL 技术,需要将 JSON 语句转换成 SQL 语句。转换过程如下:

#### 3.2.1 构建表

构建规则如下:

$url \rightarrow$ 表名

$param.name \rightarrow$ 列名, $if\ param \in$ 直接参数

$param.name\&param.value \rightarrow$ 列名, $if\ param \in$ 间接参数

根据图 5 中的 JSON 语句,可以构建出的结构见表 1。

问句:“导出 2020 年 01 月 01 日到 2020 年 12 月 31 日,西安分公司合同的台账,包含合同名称、合同金额”

JSON 语句:

```
{"method":"POST",
"url":"/api/v2/contract/query/export",
"params":[
{"name":"creation_time","option":"btw","value":["2020 年 01 月 01 日","2020 年 12 月 31 日"]},
{"name":"create_org","option":"eq","value":"西安分公司"},
{"name":"columns","option":"in","value":["contract_name","contract_amount"]}
]
}
```

**图 5 问句与 JSON 语句示例**

**Fig. 5 (question, JSON) pair**

**表 1 图 5 中 JSON 对应的数据表**

**Tab. 1 The table corresponding to JSON in Fig.5**

表名:“ */api/v2/contract/query/export* ”

| create_time | create_org | columns&contract_name | columns&contract_amount |
|---|---|---|---|
| … | … | … | … |

3.2.2　JSON 语句转换为 SQL 语句

转换规则如下所示：

$url \rightarrow table\ name$

$param \rightarrow (name, option, value), if\ param \in$ 直接参数

$param \rightarrow (option, name\&value), if\ param \in$ 间接参数

对于直接参数，转换后参数被放置在条件子句 where_clause 中；对于间接参数，转换后的参数被放置在选择子句 select_clause 中；url 被放置在 from 子句中。根据图 5 中的 JSON 语句，可以转换成如图 6 所示的 SQL 语句。

```
SQL{
SELECT:[(in,columns&constract_name),(in,columns&constract_amount)]
FROM:["/api/v2/constract/query/export"]
WHERE:[(create_time,btw,("2020 年 01 月 01 日","2020 年 12 月 31 日")),
(create_org,eq,("西安分公司"))]
}
```

**图 6　图 5 中 JSON 对应的 SQL 语句**

**Fig. 6　The SQL corresponding to JSON in Fig.5**

3.2.3　数据整合

经过从 JSON 抽取并构建表，可以得到 28 个表组成的数据库表集合；经过将 JSON 语句转换为 SQL 语句，可以得到新的2 312条（问句与 SQL 语句）对；将2 312条（问句，SQL 语句）对和数据库表集合，作为新数据集，用于模型训练、验证和测试。

## 4　实验结果

将 2 312 条数据中的 1 800 个数据对用于训练集，100 个数据对用于验证集，412 个数据对用于测试集。

表 2 展示模型使用不同语言预训练模型[27]，以及在使用执行指导和不使用执行指导的情况下，模型在验证集和测试集上的逻辑准确率。逻辑准确率是指预测的 SQL 语句和标签 SQL 语句完全一致的比例。

**表 2　不同模型在数据验证集和测试集上的逻辑准确率**

**Tab. 2　Logical form accuracy on dataset**

| 模型 | 语言预训练模型 | 验证集 | 测试集 |
|---|---|---|---|
| H-Net | RoBERTa-wwm-ext, Chinese | 84.0 | 83.5 |
| H-Net | RoBERTa-wwm-ext-large, Chinese | 85.0 | 84.7 |
| H-Net+EG | RoBERTa-wwm-ext, Chinese | 88.0 | 87.6 |
| H-Net+EG | RoBERTa-wwm-ext-large, Chinese | 89.0 | 89.1 |

参考 WikiSQL 数据集[13]和 Spider 数据集[14]的评价方式，将 SQL 语句划分为不同子 SQL 语句，分别评测模型在每类子 SQL 语句对应的子任务上的准确率。表 3 展示模型在每类子任务上、在使用和不使用执行指导的情况下，模型在验证集和测试集上逻辑准确率。

**表 3　不同模型在数据验证集和测试集上每类子任务的逻辑准确率**

**Tab. 3　Logical form accuracy of each task on dataset**

| 子任务\模型 | H-Net+EG（验证集,测试集） | H-Net（验证集,测试集） |
|---|---|---|
| S-COL | 98.7, 98.5 | 97.9, 97.6 |
| S-AGG | 98.7, 98.5 | 98.3, 98.0 |
| W-COL | 98.3, 97.9 | 98.1, 97.6 |
| W-OP | 99.2, 98.8 | 97.6, 96.8 |
| VAL_NUM | 98.7, 98.5 | 96.2, 95.7 |
| VAL | 96.2, 96.0 | 95.7, 93.7 |
| O-COL | 99.2, 98.8 | 98.9, 98.6 |
| O-OR | 99.4, 99.3 | 99.2, 98.7 |
| Rel | 99.8, 99.6 | 99.4, 99.1 |
| Link | 99.8, 99.6 | 99.4, 99.1 |
| Tem | 99.8, 99.6 | 99.4, 99.1 |

## 5　结束语

本文研究利用语言预训练模型和深度学习来进行文本到指令序列的翻译，提出了一种混合排序填充网络模型，该模型可以很好地利用语言预训练模型的特点，并且能处理复杂的 SQL 语句。经在自构建的文本到指令序列数据集上测试，取得了很好的实验结果，翻译结果的逻辑准确率可以达到 89.1%。

## 参考文献


[1] Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. Natural language interfaces to databases – an introduction. Natural language engineering, 1995, 1(1): 29-81.

[2] 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.

[3] WARREN D H D, PEREIRA F C N. An efficient easily adaptable system for interpreting natural language queries[J]. American journal of computational linguistics, 1982, 8(3-4): 110-122.
[4] WOODS W A. Readings in natural language processing. chapter Semantics and quantification in natural language question answering[J]. 1986.
[5] HENDRIX G G, SACERDOTI E D, SAGALOWICZ D, et al. Developing a natural language interface to complex data[J]. ACM Transactions on Database Systems (TODS), 1978, 3(2): 105-147.
[6] GROSZ B J, APPELT D E, MARTIN P A, et al. TEAM: An experiment in the design of transportable natural - language interfaces[J]. Artificial Intelligence, 1987, 32(2): 173-243.
[7] ANDROUTSOPOULOS I, RITCHIE G, THANISCH P. Masque/sql ¦ An E cient and Portable Natural Language Query Interface for Relational Databases[J]. Database technical paper, Department of AI, University of Edinburgh, 1993.
[8] TANG L R, MOONEY R. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing [C]//2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 2000: 133-141.
[9] CHO K, VAN MERRIËNBOER B, GULCEHRE C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[J]. arXiv preprint arXiv: 1406.1078, 2014.
[10] SUTSKEVER I, VINYALS O, LE Q V. Sequence to sequence learning with neural networks[J]. arXiv preprint arXiv: 1409.3215, 2014.
[11] GUO J, ZHAN Z, GAO Y, et al. Towards complex text-to-sql in cross-domain database with intermediate representation[J]. arXiv preprint arXiv:1905.08205, 2019.
[12] WANG B, SHIN R, LIU X, et al. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers[J]. arXiv preprint arXiv:1911.04942, 2019.
[13] ZHONG V, XIONG C, SOCHER R. Seq2sql: Generating structured queries from natural language using reinforcement learning[J]. arXiv preprint arXiv:1709.00103, 2017.
[14] 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[J]. arXiv preprint arXiv:1809.08887, 2018.
[15] Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.
[16] DONG L, LAPATA M. Coarse-to-fine decoding for neural semantic parsing[J]. arXiv preprint arXiv:1805.04793, 2018.
[17] YIN P, NEUBIG G. A syntactic neural model for general-purpose code generation[J]. arXiv preprint arXiv:1704.01696, 2017.
[18] 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.
[19] XU X, LIU C, SONG D. Sqlnet: Generating structured queries from natural language without reinforcement learning[J]. arXiv preprint arXiv:1711.04436, 2017.
[20] YU T, LI Z, ZHANG Z, et al. Typesql: Knowledge-based type-aware neural text-to-sql generation[J]. arXiv preprint arXiv: 1804.09769, 2018.
[21] HWANG W, YIM J, PARK S, et al. A comprehensive exploration on wikisql with table-aware word contextualization [J]. arXiv preprint arXiv:1902.01069, 2019.
[22] HE P, MAO Y, CHAKRABARTI K, et al. X-SQL: reinforce context into schema representation [J]. Microsoft Research: Artificial Intelligence, 2019.
[23] LYU Q, CHAKRABARTI K, HATHI S, et al. Hybrid ranking network for text-to-sql[J]. arXiv preprint arXiv: 2008.04759, 2020.
[24] Sepp Hochreiter, Jürgen Schmidhuber. Long Short-Term Memory [J]. Neural Comput 1997; 9 (8): 1735-1780.
[25] Tao Yu and Chien-Sheng Wu and Xi Victoria Lin and Bailin Wang and Yi Chern Tan and Xinyi Yang and Dragomir Radev and Richard Socher and Caiming Xiong. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing[J]. arxVix eprint: 2009.13845, 2020.
[26] WANG C, TATWAWADI K, BROCKSCHMIDT M, et al. Robust text-to-sql generation with execution-guided decoding [J]. arXiv preprint arXiv:1807.03100, 2018.
[27] CUI Y, CHE W, LIU T, et al. Pre-training with whole word masking for chinese bert[J]. arXiv preprint arXiv: 1906.08101, 2019.

(上接第 108 页)

地对该机器人的性能进行评价。该机器人的位置准确度为 1.021 mm,大于一般的工业标准 0.02 mm[8],因此在利用该机器人进行实验时,需要先进行 DH 参数校准。其位姿、距离和轨迹的重复性均小于准确度,表明该机器人可以通过 DH 参数校准,提升各项指标的准确度。

## 参考文献

[1] 陈刚,吴雪珂,欧永,等. 激光跟踪仪在机器人性能测试中的应用[J]. 电子产品可靠性与环境试验, 2018, 36(3):61-69.
[2] 李新,茅晨,马涛,等. 利用 Leica 激光跟踪仪对工业机器人现场标定的方法[J]. 计量技术, 2019(11):64-68.
[3] 乔贵方,孙大林,宋光明,等. 串联机器人标定系统的坐标系快速转换方法[J]. 机械工程学报, 2020, 56(14):1-8.
[4] 孙大林,乔贵方,宋光明,等. 应用于机器人标定的主动式靶标装置设计与精度优化[J]. 测控技术,2019. 38(8):11-14,36.
[5] 任瑜,张丰,郭志敏. 一种通用的工业机器人位姿检测方法[J]. 计量学报,2018, 39(5):615-621.
[6] 贺惠农,黄连生. 工业机器人整机测试性能进展[J]. 中国计量大学学报,2017, 28(2):133-140.
[7] 中华人民共和国国家标准,GB/T 12642-2013,工业机器人性能规范及其试验方法[S]. 北京:中国标准出版社,2014.
[8] 杭强,施威涛. 基于激光跟踪仪的机器人性能测量与分析[J]. 中国计量, 2019(1):117-118.