赛尔笔记 | TEXT-TO-SQL任务介绍

作者：窦隆绪、潘名扬、乔振浩

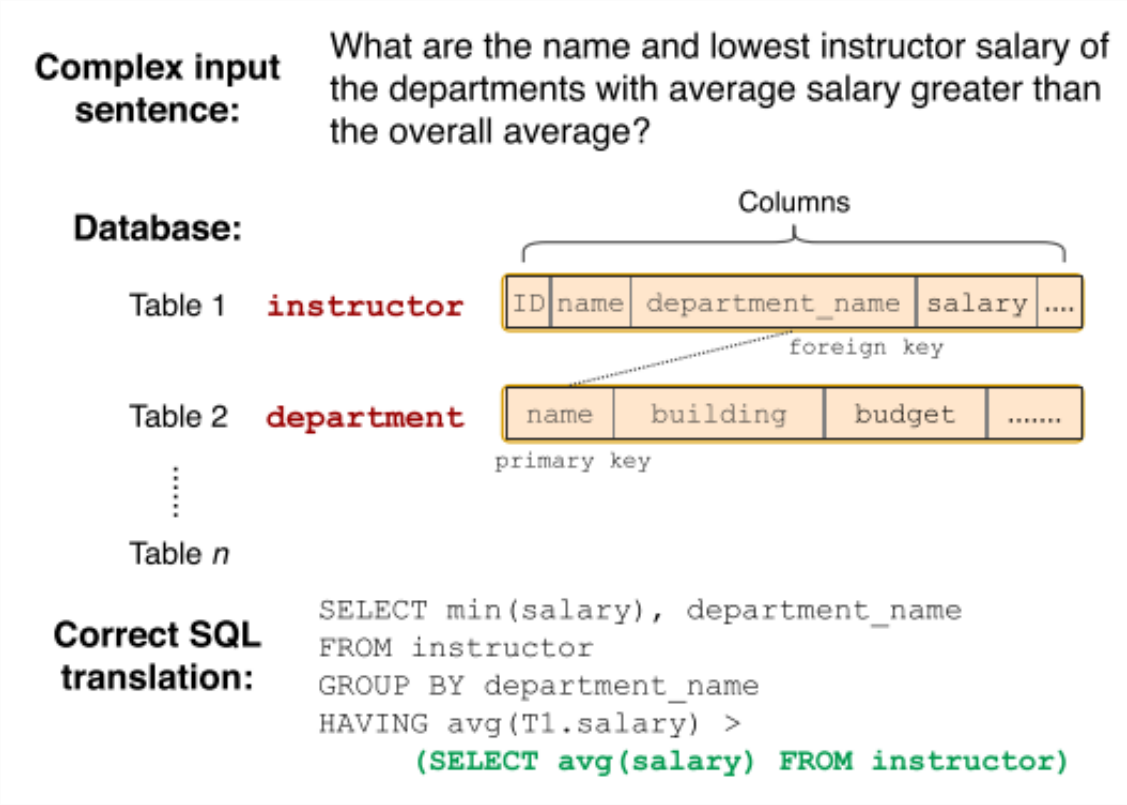
任务介绍

在现实生活中有许许多多的数据库，存储着各行各业的信息，比如学校的选课信息、成绩信息，公司的账务信息、人员流动。而SQL作为一种专门的数据库语言，对于普通人来说，学习门槛比较高。即使对于计算机从业者来说，想要针对不同的数据库和应用场景，编写合适且正确的SQL语句，也着实需要下一番功夫。如果我们能够作出一个工具，自动的把我们的需求转化为查询语句，再交给计算机去执行，使得所有人都能方便的进行查询，那就大大提高了我们的生活和工作效率。

因此Text-to-SQL是一个非常具有实用价值的任务。举个例子：当我们询问智能助手“李安导演是在哪出生的啊？”，Text-to-SQL模型就会先根据问句解析出SQL语句-“SELECT birth_city FROM director WHERE name = "李安"”，再向用户返回查询结果-“台湾屏东县潮州镇”。

目前这个研究问题引起了学术界和工业界的广泛关注，有着很多比赛榜单和标注数据集，比较有名的包括早起的订票系统ATIS、地理数据库GeoQuery、基于维基百科的WikiSQL以及耶鲁大学发布的目前最复杂的Spider。在中文Text-to-SQL任务方面，西湖大学日前公布了中文版的Spider数据集，追一科技在天池数据平台举行了第一届中文text2sql挑战赛。

Text-to-SQL任务更为正式的定位为：在指定关系型数据库（或表）的前提下，由用户的提问生成相应的SQL查询语句。用户的提问可以是上下文无关的，例如：



也可以是上下文关联的：

Database: student dormitory containing 5 tables.

Goal: Find the first and last names of the students who are living in the dorms that have a TV Lounge as an amenity.

Q1: How many dorms have a TV Lounge?

S1:
```
SELECT COUNT(*) FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3
ON T2.amenid = T3.amenid WHERE T3.amenity_name = 'TV Lounge'
```

Q2: What is the total capacity of these dorms?

S2:
```
SELECT SUM(T1.student_capacity) FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN
dorm_amenity AS T3 ON T2.amenid = T3.amenid WHERE T3.amenity_name = 'TV Lounge'
```

Q3: How many students are living there?

S3:
```
SELECT COUNT(*) FROM student AS T1 JOIN lives_in AS T2 ON T1.stuid = T2.stuid WHERE T2.dormid
IN (SELECT T3.dormid FROM has_amenity AS T3 JOIN dorm_amenity AS T4 ON T3.amenid = T4.amenid
WHERE T4.amenity_name = 'TV Lounge')
```

Q4: Please show their first and last names.

S4:
```
SELECT T1.fname, T1.lname FROM student AS T1 JOIN lives_in AS T2 ON T1.stuid = T2.stuid WHERE T2.dormid
IN (SELECT T3.dormid FROM has_amenity AS T3 JOIN dorm_amenity AS T4 ON T3.amenid = T4.amenid
WHERE T4.amenity_name = 'TV Lounge')
```

在本篇文章中，我们首先会对SQL生成的相关数据集进行介绍，之后介绍Spider（上下文无关、跨领域）数据集上的主流模型。

问题分析

Text-to-SQL在某种程度上可以看做是一个翻译任务，也就是将SQL当作一门外语。于是，最简单也最粗暴的做法就是直接采用一个seq2seq的模型。在加上Attention、Copying等机器翻译常用的机制之后，该方法能够在WikiSQL上能达到23.3%的精确匹配，37.0%的执行正确率；在Spider上则只能达到5～6%的准确率。

究其原因，可以从编码和解码两个角度来看。首先编码方面，自然语言问句与数据库之间需要形成很好的对齐或映射关系，即问题中到底涉及了哪些表格中的哪些实体词，以及问句中的词语触发了哪些选择条件、聚类操作等；另一方面在解码部分，SQL作为一种形式语言，本身对语法的要求更严格（关键字顺序固定）以及语义的界限更清晰，失之毫厘差之千里。

于是，主流模型的改进与后续工作无不围绕着以下两个方面展开：通过更强的表示（BERT、XLNet）、更优秀的结构信息（GNN）、更多的数据（数据增强）来加强Encoder端的对齐关系；通过树形结构解码、填槽类解码以及解码后的重排序来减小搜索解空间，以增加SQL语句的正确性。

相关数据集介绍

1.ATIS

ATIS（Air Travel Information System），是关于航班信息的、由用户提问生成SQL语句的单一领域、上下文相关的数据集，采用了BIO标注的方式来标识问句中所包含的的表名和查询条件等。

2.GeoQuery

是与地理区域相关的，单一领域、上下文无关的数据集。特点：按照问句所表达的含义进行分组，每一组表示同一类查询，每组提供几个SQL语句。

3.SequentialQA

SequentialQA是跨领域的，上下文相关的数据集。对于每一个表，给数据集给出了三组上下文相关的提问，并给出了对应的查询结果（没有直接给出SQL语句）。

4.WikiSQL

WikiSQL是跨领域的，上下文无关的数据集。来源于英文维基百科，数据规模较大。但是SQL语句结构较为简单，不需要进行选取表的操作，Where子句的条件使用and连接。

5.Spider

Spider是跨领域的，上下文无关的数据集。领域比较丰富（拥有来自138个领域的200多个数据库），并且训练集、测试集中出现的数据库不重合。SQL语句更为复杂，包含orderBy、union、except、groupBy、intersect、limit、having 关键字，以及嵌套查询等。并且该数据集依据SQL语句的复杂程度，将数据分为了四个难度等级。下面是Hard和Extra Hard的实例：

Hard

Which countries in Europe have at least 3 car manufacturers?

```
SELECT T1.country_name
FROM countries AS T1 JOIN continents
AS T2 ON T1.continent = T2.cont_id
JOIN car_makers AS T3 ON
T1.country_id = T3.country
WHERE T2.continent = 'Europe'
GROUP BY T1.country_name
HAVING COUNT(*) >= 3
```

Extra Hard

What is the average life expectancy in the countries where English is not the official language?

```
SELECT AVG(life_expectancy)
FROM country
WHERE name NOT IN
    (SELECT T1.name
     FROM country AS T1 JOIN
     country_language AS T2
     ON T1.code = T2.country_code
     WHERE T2.language = "English"
       AND T2.is_official = "T")
```

6.Sparc

SParC是跨领域的，上下文相关的数据集。数据库基于Spider，并且模拟了用户进行数据库查询的过程：用户通过若干条相关的提问最后达到一个最终查询目的。

不同数据集对比：

DATASET	#Q	#SQL	#DB	#DOMAIN	#TABLE/DB	ORDERBY	GROUPBY	NESTED	HAVING	CONTEXT	RESOURCE	ANNOTATION
ATIS	5,280	947	1	1	32	0	5	315	0	√	database	SQL
GEOQUERY	877	247	1	1	6	20	46	167	9	×	database	SQL
SEQUENTIALQA	17,555	-	982	many	1	-	-	-	-	√	table	denotation
WIKISQL	80,654	77,840	26,521	many	1	0	0	0	0	×	table	SQL
SPIDER	10,181	5,693	200	138	5.1	1335	1491	844	388	×	database	SQL
SPARC	12,726	12,726	200	138	5.1	2087	2467	700	577	√	database	SQL

Spider Benchmark（截止12.19日）：

Rank	Model	Dev	Test
1 Dec 13, 2019	RATSQL v2 + BERT (DB content used) *Anonymous*	65.8	**61.9**
2 Dec 18, 2019	IRNet++ + XLNet (DB content used) *Anonymous*	65.5	60.1
3 Nov 12, 2019	RYANSQL + BERT *Anonymous*	66.6	58.2
4 Dec 13, 2019	RATSQL v2 (DB content used) *Anonymous*	62.7	57.2
5 Dec 13, 2019	RASQL + BERT *Anonymous*	60.8	55.7
6 Dec 13, 2019	EditSQL+LSL + BERT *Anonymous*	57.9	55.2
7 June 24, 2019	IRNet v2 + BERT *Microsoft Research Asia*	63.9	55.0
8 Sep 20, 2019	GIRN + BERT *Anonymous*	60.2	54.8
9 May 19, 2019	IRNet + BERT *Microsoft Research Asia* (Guo and Zhan et al., ACL '19) code	61.9	54.7
10 Nov 4, 2019	BERTRAND + GNN *Anonymous*	57.9	54.6

主流工作

1.Pointer Network

传统的seq2seq模型的解码器部分所使用的单词表是固定的，即在生成序列中都是从固定的单词表中进行选取。但Text-to-SQL不同于一般的seq2seq任务，它的生成序列中可能出现：a）问句中的单词; b）SQL关键字; c)对应数据库中的表名、列名。

Pointer Network很好地解决了这一问题，其输出所用到的词表是随输入而变化的。具体做法是利用注意力机制，直接从输入序列中选取单词作为输出。

在Text-to-SQL任务中，可以考虑把用户的提问以及目标SQL语句可能出现的其他词作为输入序列 (列名单词序列；SQL的关键字表；问题的单词序列)，利用Pointer Network直接从输入序列中选取单词作为输出。在解码器的每一步，与编码器的每一个隐层状态计算注意力分数，取最大值作为当前的输出以及下一步的输入。

2.Seq2SQL

Pointer Network虽然一定程度上解决了问题，但是它并没有利用到SQL语句固有的语法结构。Seq2SQL将生成的SQL语句分为三个部分： 聚合操作： （SUM、COUNT、MIN、MAX等）、SELECT：选取列、WHERE：查询条件。每一部分使用不同的方法进行计算。

SELECT与聚合操作，均采用了注意力机制进行分类。WHERE子句可以利用前面介绍的Pointer Network进行训练，但是对于很多查询来说，WHERE子句的写法并不是唯一的，例如：

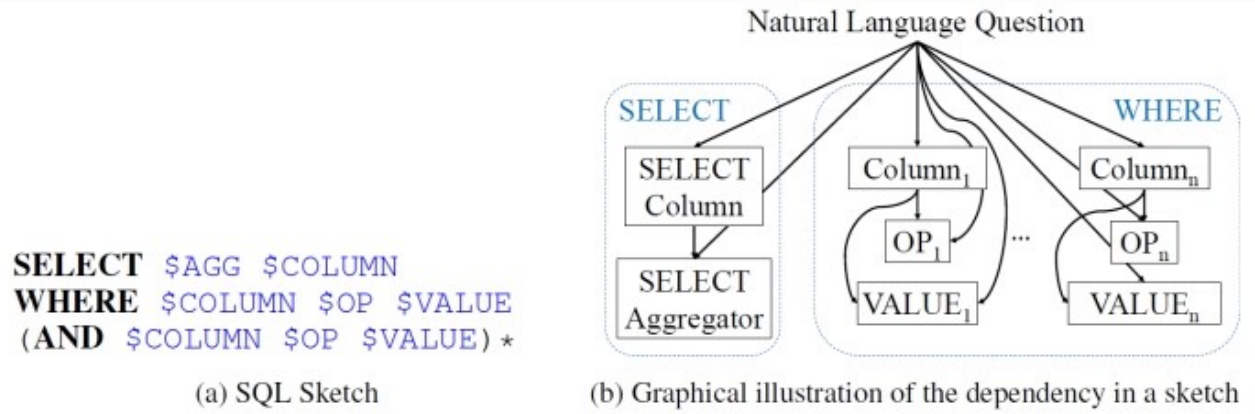
SELECT name FROM insurance WHERE age > 18 AND gender = "male";

SELECT name FROM insurance WHERE gender = "male" AND age > 18;

这可能导致原本正确的输出被判断为错误的。于是作者提出利用强化学习基于查询结果来进行优化。在解码器部分，对可能的输出进行采样，产生若干个SQL语句，每一句表示为y=(y1,y2...yT)，用打分函数对每一句进行打分：

$$R\left(q\left(y\right),q_g\right)=\begin{cases}-2, & \text{if } q\left(y\right) \text{ is not a valid SQL query}\\ -1, & \text{if } q\left(y\right) \text{ is a valid SQL query and executes to an incorrect result}\\ +1, & \text{if } q\left(y\right) \text{ is a valid SQL query and executes to the correct result}\end{cases}$$

3.SQLNet



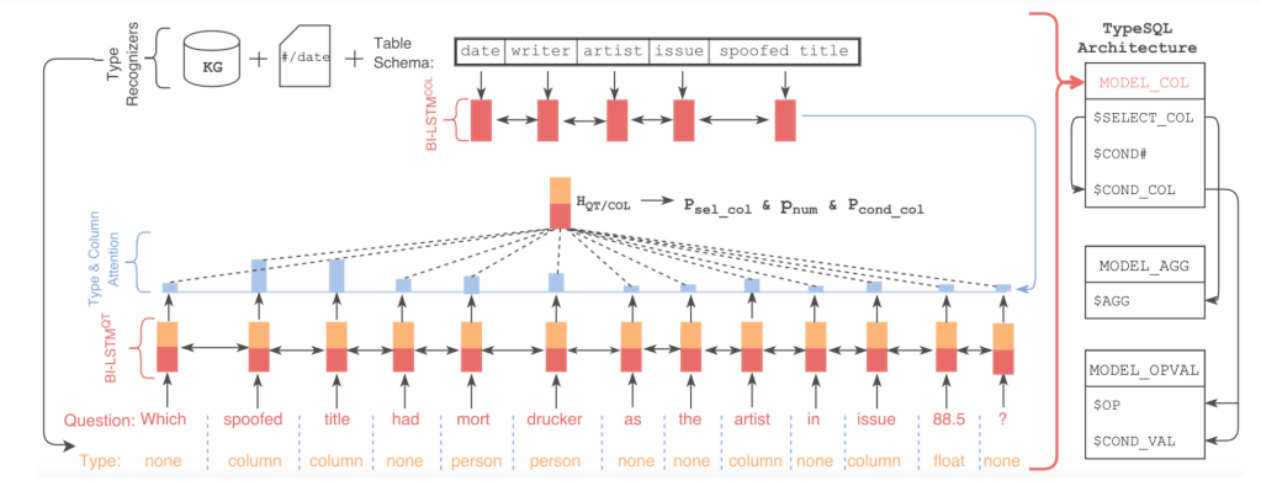
(a) SQL Sketch (b) Graphical illustration of the dependency in a sketch

Figure 2: Sketch syntax and the dependency in a sketch

为了解决Seq2SQL使用强化学习效果不明显的问题，SQLNet将SQL语句分成了SELECT和WHERE两个部分，每个部分设置了几个槽位，只需向槽位中填入相应的符号即可。

SELECT子句部分与Seq2SQL类似，不同地方在于WHERE子句，它使用了一种sequence-to-set（由序列生成集合）机制，用于选取目标SQL语句中的WHERE子句可能出现的列。对于表中的每一列给出一个概率。之后计算出WHERE子句中的条件个数k，然后选取概率最高的前k个列。最后通过注意力机制进行分类得到操作符和条件值。

4.TypeSQL



该模型基于SQLNet，使用模版填充的方法生成SQL语句。为了更好地建模文本中出现的罕见实体和数字，TypeSQL显式地赋予每个单词类型。

类型识别过程：将问句分割n-gram （n取2到6），并搜索数据库表、列。对于匹配成功的部分赋值column类型赋予数字、日期四种类型：INTEGER、FLOAT、DATE、YEAR。对于命名实体，通过搜索FREEBASE，确定5种类型：PERSON，PLACE，COUNTREY，ORGANIZATION，SPORT。这五种类型包括了大部分实体类型。当可以访问数据库内容时，进一步将匹配到的实体标记为具体列名（而不只是column类型）

SQLNet为模版中的每一种成分设定了单独的模型；TypeSQL对此进行了改进，对于相似的成分，例如$SELECT_COL 和$COND_COL以及#COND（条件数），这些信息间有依赖关系，通过合并为了单一模型，可以更好建模。TypeSQL使用3个独立模型来预测模版填充值：

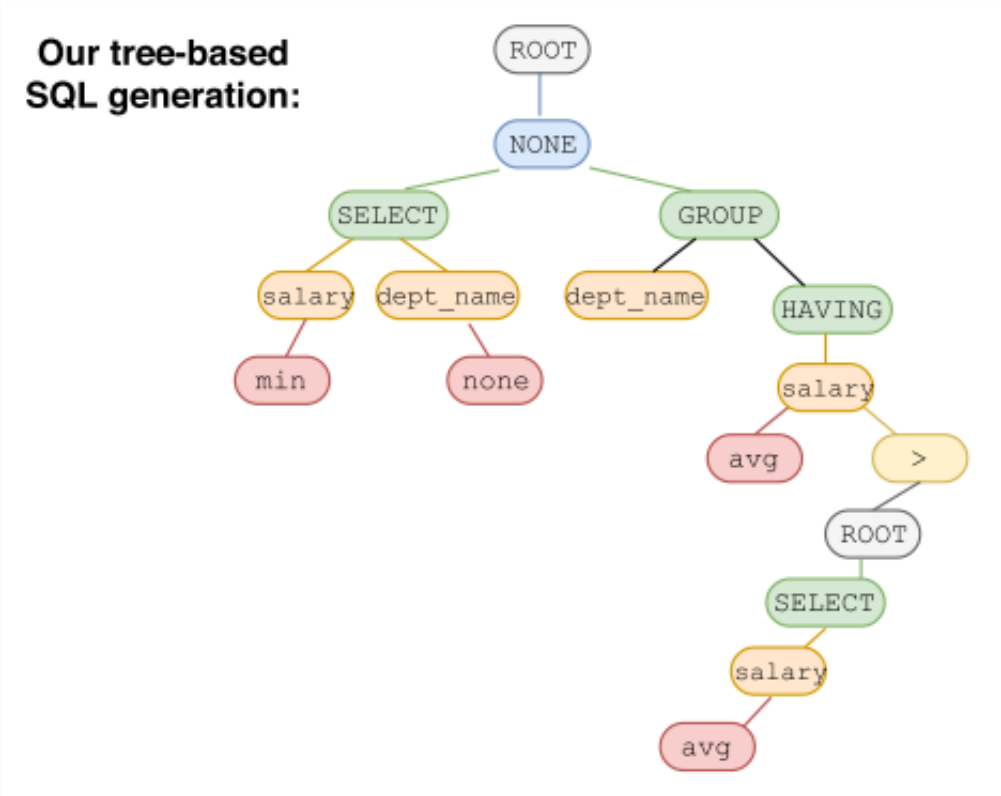
MODEL_COL：$SELECT_COL，$COND#，$COND_COL

MODEL_AGG：$AGG

MODEL_OPVAL：$OP, $COND_VAL

5.SyntaxSQLNet

相比于之前decoder输出一段线性的文本，SyntaxSQLNet将解码的过程引入了结构性信息，即解码的对象为SQL语句构成的树结构。（准确率+14.8%）



SyntaxSQLNet将SQL语句的预测分解为9个模块，每个模块对应了SQL语句中的一种成分。解码时由预定义的SQL文法确定这9个模块的调用顺序，从而引入结构信息。树的生成顺序为深度优先。分解出的9个模块有：

IUEN模块：预测INTERCEPT、UNION、EXCEPT、NONE（嵌套查询相关）

KW模块：预测WHERE、GROUP BY、ORDER BY、SELECT关键字

COL模块：预测列名

OP模块：预测>、<、=、LIKE等运算符

AGG模块：预测MAX、MIN、SUM等聚合函数

Root/Terminal模块：预测子查询或终结符

Module模块：预测子查询或终结符

AND/OR模块：预测条件表达式间的关系

DESC/ASC/LIMIT模块：预测与ORDER BY相关联的关键字

HAVING模块：预测与GROUP BY相关的Having从句

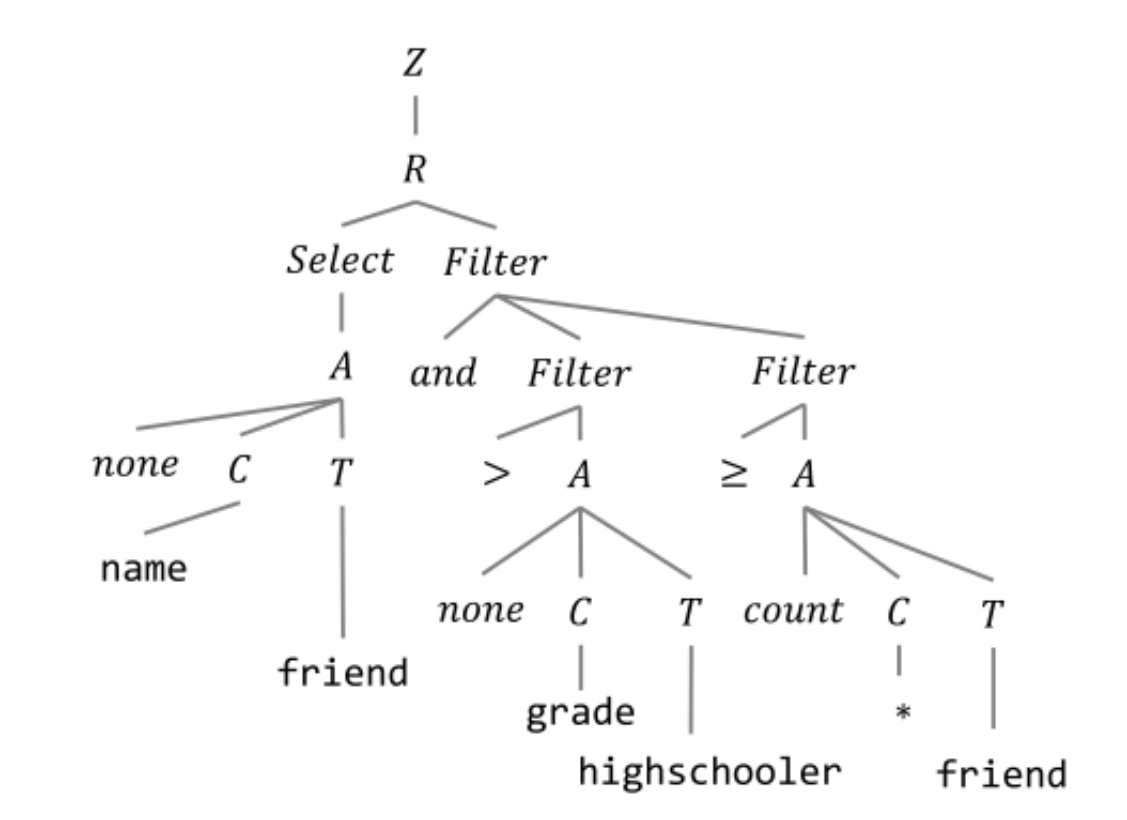
该工作同时提供了一种针对text2sql任务的数据增强方法，生成跨领域、更多样的训练数据。（准确率++7.5%）

具体做法为：对SPIDER中的每条数据，将值和列名信息除去，得到一个模版； 对处理后的SQL模版进行聚类，通过规则去除比较简单的模版，并依据模板出现的频率，挑选50个复杂SQL模板； 人工核对SQL-问句对，确保SQL模板中每个槽在问句中都有对应类型的信息。

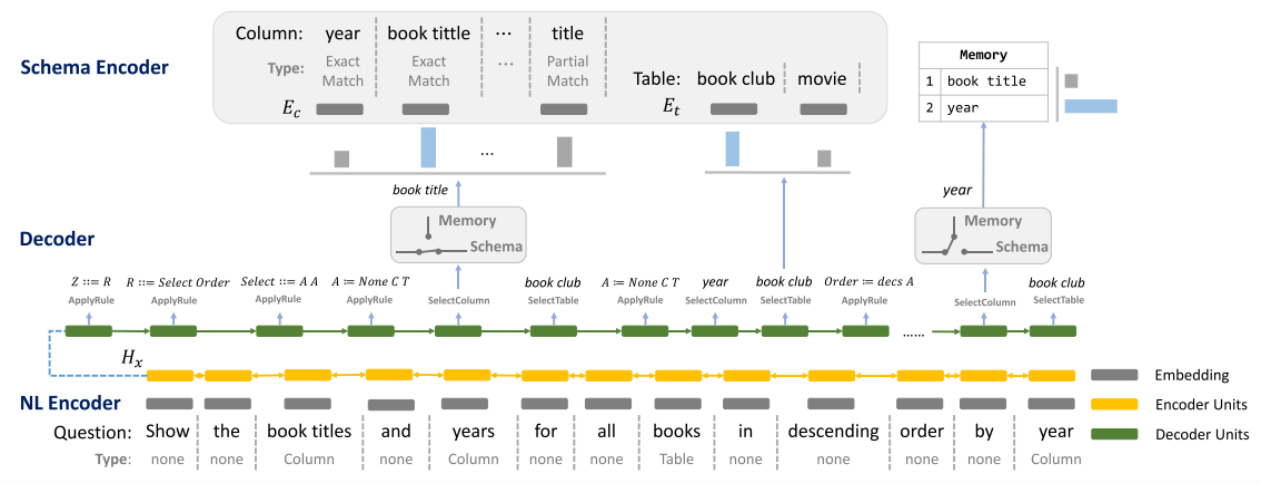
得到一一对应的模板后，应用于WikiSQL数据库：首先随机挑选10个模板，然后从库中选择相同类型的列，最后用列名和值填充SQL模板和问句模板。通过该方法，作者最终在18000的WikiSQL数据库上得到了新的98000组训练数据，同时在训练的时候也利用了WikiSQL数据集原有的训练数据。

6.IRNet

与SyntaxSQLNet类似，IRNet定义了一系列的CFG文法，将SQL转发为语法树结构。可以将其看作一种自然语言与SQL语句间的中间表示（作者称之为SemQL），整个parsing的过程也是针对SemQL进行的。如下：

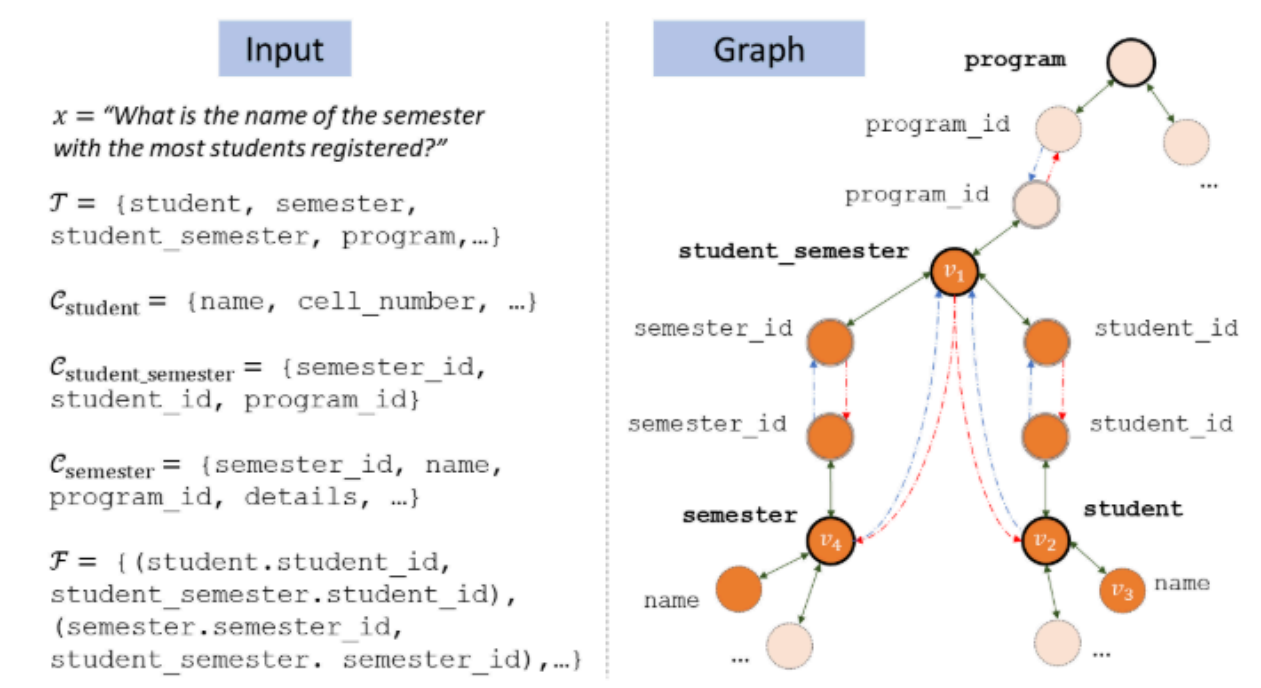


作者另一部分的改进主要在scheme linking，即如何找到问题中所提到的表格与列。他将问题中可能出现的实体分为3类：表格名、列名、表中的值。根据3类实体的不同，具体做法分为：a）表格名和列名：以n-gram的形式枚举问题中的span，然后和表格名、列名进行匹配。可以看到下图中的Question中对应的单词有的被标成了Column或者Table。 b) 表中的值：将问题中以引号为开头结尾的span，送给conceptnet进行查询，再将返回结果中的 `is a type of'/'related terms'关系的词与列名进行匹配。

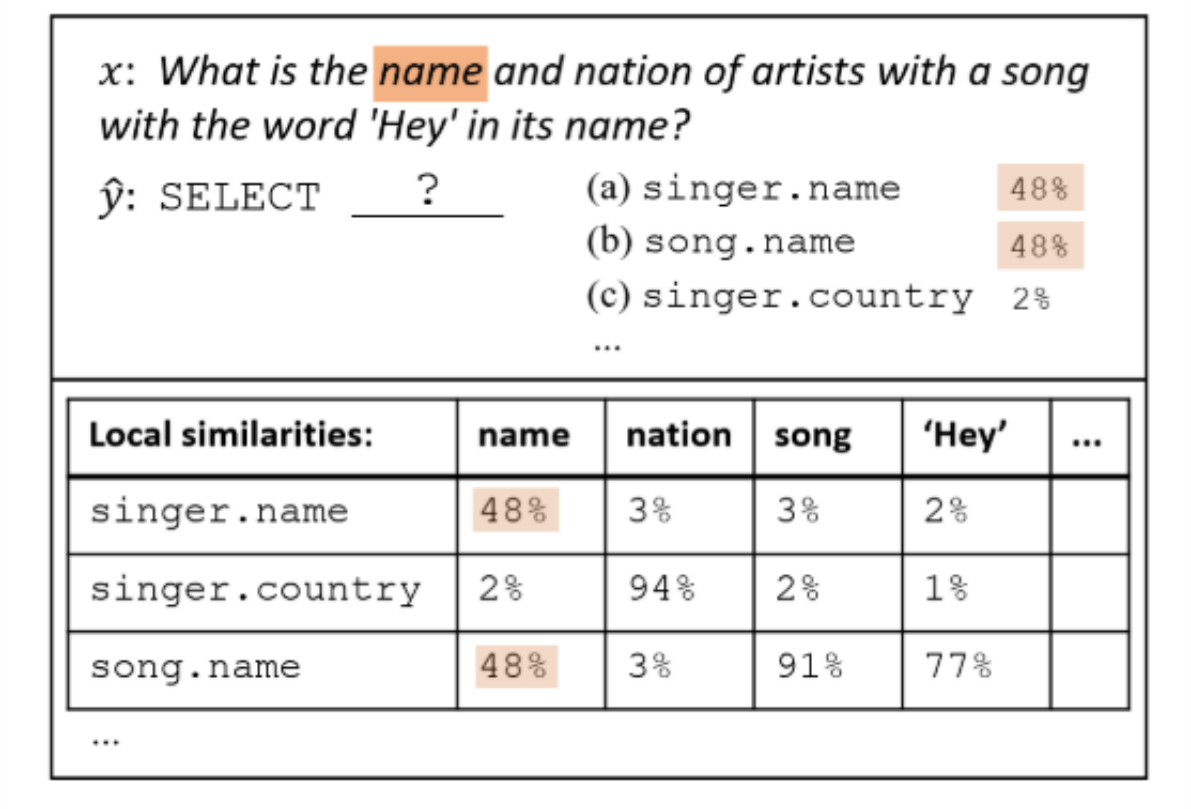


7.Global-GNN

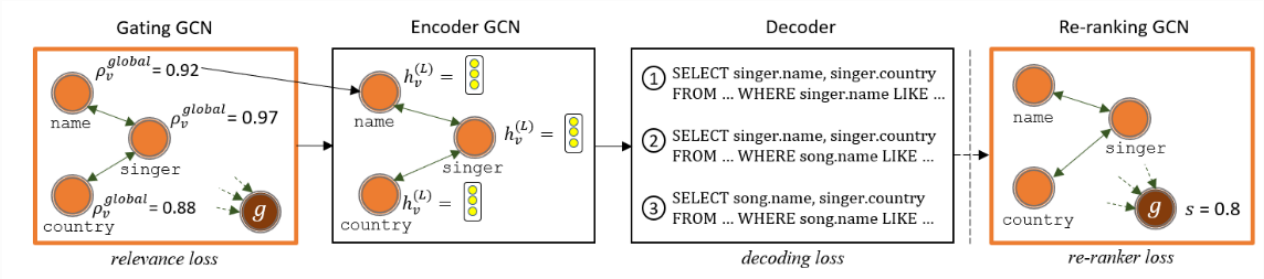
为了更好的利用关系型数据库的结构信息，Ben Bogin等研究人员提出使用图网络来建模表格名和列名。如下图所示：圆圈加粗的结点代表表格，不加粗的结点代表列名；双向边代表表格和列名的从属关系；红虚边和蓝虚边代表主外键关系。橙色节点代表与问题有关的结果，淡色为无关。



除此之外，该团队还提出了一种基于全局信息重排序的做法。首先先看下面这个例子，我们不知道name到底指向的是singer还是song，但是我们可以观察到nation只在singer中出现，所以应该是singer.name。这样做global reasoning，就能减小歧义性。

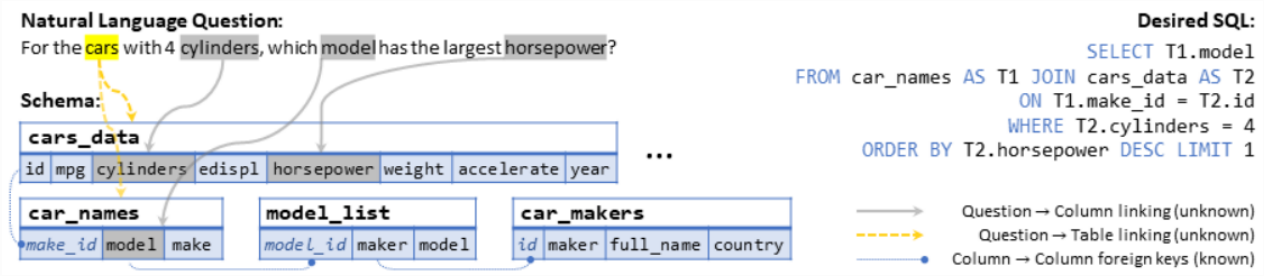


整个工作的流程如下图所示：



8.RAT-SQL

该工作可以看做图网络建模的后续工作，在Table、Column、Quetion三者之间定义了更多的边（共33种），是目前榜单上的最强模型。



Type of x	Type of y	Edge label	Description
Column	Column	SAME-TABLE	x and y belong to the same table.
		FOREIGN-KEY-COL-F	x is a foreign key for y.
		FOREIGN-KEY-COL-R	y is a foreign key for x.
Column	Table	PRIMARY-KEY-F	x is the primary key of y.
		BELONGS-TO-F	x is a column of y (but not the primary key).
Table	Column	PRIMARY-KEY-R	y is the primary key of x.
		BELONGS-TO-R	y is a column of x (but not the primary key).
Table	Table	FOREIGN-KEY-TAB-F	Table x has a foreign key column in y.
		FOREIGN-KEY-TAB-R	Same as above, but x and y are reversed.
		FOREIGN-KEY-TAB-B	x and y have foreign keys in both directions.

所有模型在WikiSQL和Spider的表现如下图所示

Model	WikiSQL						Spider					
	Dev			Test			Test(Acc em)					Dev(Acc em)
	Acc ex	Acc em	Acc qm	Acc ex	Acc em	Acc qm	Easy	Medium	Hard	Extra Hard	All	All
Ptr Network	53.80%	44.10%	-	53.30%	43.30%	-	-	-	-	-	-	-
Seq2SQL	60.80%	49.50%	-	59.40%	48.30%	-	-	-	-	-	-	-
SQLNet	69.80%	-	63.20%	68.00%	-	61.30%	26.20%	12.60%	6.60%	1.30%	12.40%	10.90%
TypeSQL	74.50%	-	68.00%	73.50%	-	66.70%	19.60%	7.60%	3.80%	0.80%	8.20%	8.00%
TypeSQL+table content	85.50%	-	79.20%	82.60%	-	75.40%	-	-	-	-	-	-
SyntaxSQLNet	-	-	-	-	-	-	48.00%	27.00%	24.30%	4.60%	27.20%	24.80%
IRNet	-	-	-	-	-	-	70.10%	49.20%	39.50%	19.10%	46.70%	53.20%
Global-GNN	-	-	-	-	-	-	-	-	-	-	47.40%	52.10%
RAT-SQL	-	-	-	-	-	-	73.10%	60.10%	45.30%	24.80%	53.70%	60.60%

总结

目前，SQL生成任务因其实用的应用场景，引起了学术界和工业界的广泛关注。目前大家的做法也是百花齐放：中间表示、树形解码、图网络建模Quetion和数据库间的关系、重排序、数据增强。但目前的模型，还不能很好解决复杂的操作，例如IRNet在Hard和Extra Hard的准确率也仅为48.1%和25.3%。期待后面能有更加有效、简洁、优雅的工作出现。

相关引用：

1. Seq2sql: Generating structured queries from natural language using reinforcement learning
2. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation

3. SParC: Cross-Domain Semantic Parsing in Context
4. CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases
5. Pointer Networks
6. Neural semantic parsing with type constraints for semi-structured tables
7. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning
8. TypeSQL: Knowledge-based Type-Aware Neural Text-to-SQL Generation
9. SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-DomainText-to-SQL Task
10. Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions
11. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation
12. Global Reasoning over Database Structures for Text-to-SQL Parsing
13. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers
14. Learning to Map Context-Dependent Sentences to Executable Formal Queries