1. 前言
2. 数据集
3. 研究方法
4. 传统方法 基于规则
5. 深度学习方法
6. Query without sql
7. Select agg where
8. X-sql
9. 对X-SQL的改进   
   因为某些原因，X-sql 跟数据集不匹配，所以要改进。
10. 展望

生成的可视化图形

能够让用户快速找到与问题对应的答案，在不需要重新提问的前提下，还能发现与之相关的信息

。

1前言

2数据集

3研究方法

4传统方法 基于规则和统计

5深度学习方法

6Query without sql

7Select agg where

8X-sql

9对X-SQL的改进   
 因为某些原因，X-sql 跟数据集不匹配，所以要改进。

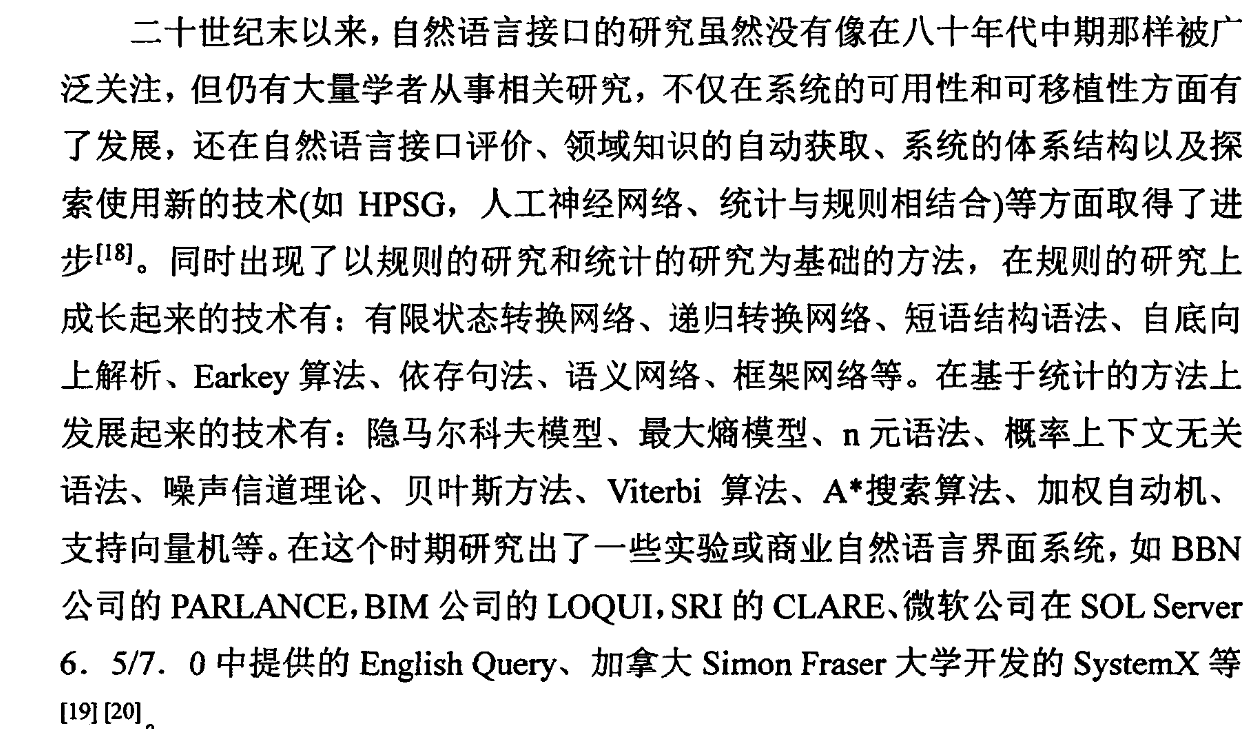
1. 展望

随着信息技术的快速变革，如何通过自然语言 直接与传统关系数据库交互，也就是如何将人的自 然问句转化成数据库可以执行的 SQL 语句已经成 为人工智能领域最热门的研究热点之一，特别是在金融领域的应用更加的广泛，为了方便管理，我们可以利用关系型数据库来存储数据，结构化查询语言(Structured Query Language)简称SQL，是一种特殊目的的编程语言，是一种数据库查询和[程序设计语言](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1%E8%AF%AD%E8%A8%80/2317999" \t "https://baike.baidu.com/item/%E7%BB%93%E6%9E%84%E5%8C%96%E6%9F%A5%E8%AF%A2%E8%AF%AD%E8%A8%80/_blank)，用于存取数据以及查询、更新和管理[关系数据库系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%95%B0%E6%8D%AE%E5%BA%93%E7%B3%BB%E7%BB%9F" \t "https://baike.baidu.com/item/%E7%BB%93%E6%9E%84%E5%8C%96%E6%9F%A5%E8%AF%A2%E8%AF%AD%E8%A8%80/_blank)。

结构化查询语言是高级的非过程化编程语言，允许用户在高层[数据结构](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84/1450" \t "https://baike.baidu.com/item/%E7%BB%93%E6%9E%84%E5%8C%96%E6%9F%A5%E8%AF%A2%E8%AF%AD%E8%A8%80/_blank)上工作。它不要求用户指定对数据的存放方法，也不需要用户了解具体的数据存放方式，所以具有完全不同底层结构的不同[数据库系统](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%B3%BB%E7%BB%9F/215176" \t "https://baike.baidu.com/item/%E7%BB%93%E6%9E%84%E5%8C%96%E6%9F%A5%E8%AF%A2%E8%AF%AD%E8%A8%80/_blank), 可以使用相同的结构化查询语言作为数据输入与管理的接口。结构化查询语言语句可以嵌套，这使它具有极大的灵活性和强大的功能。

然而， 通过关系型数据库来查询数据需要预先学习数据 库系统以及数据库查询语句 SQL 上的专业知识，这 对一个普通用户来说显然很难实现。为了解决这个 问题，许多研究工作都采用了相似的解决思路，通 过实现一个自然语言查询接口（NLI），来完成用户 通过日常使用的自然语言与数据库迚行交互从而 获取想要查询的数据[1]。

曾经有段时间落寞了，因为图形化的接口太好用了。



国内外研究现状

如今，自然语言查询接口也产生了很多商业上 的应用，例如智能语音助手、智能搜索引擎和对话 系统等。 NLI 最重要的一个部分就是将自然语言查询 描述转化为标准的 SQL 查询语句（NL2SQL）。

针 对这个问题，目前有几个不同的数据集：

we release WikiSQL, a corpus of 80654 hand-annotated instances of natural language questions, SQL queries, and SQL tables extracted from 24241 HTML tables from Wikipedia. WikiSQL is an order of magnitude larger than previous semantic parsing datasets that provide logical forms along with natural language utterances. We release the tables used in WikiSQL both in raw JSON format as well as in the form of a SQL database. Along with WikiSQL, we release a query execution engine for the database used for in-the-loop query execution to learn the policy. On WikiSQL, Seq2SQL outperforms a previously state-of-the-art semantic parsing model by Dong



WikiSQL is a collection of questions, corresponding SQL queries, and SQL tables. A single example in WikiSQL, shown in Figure 2, contains a table, a SQL query, and the natural language question corresponding to the SQL query. Table 1 shows how WikiSQL compares to related datasets. Namely, WikiSQL is the large sthand-annotated semantic parsing dataset to date - it is an order of magnitude larger than other datasets that have logical forms, either in termsofthenumberofexamplesorthenumber of tables. The queries in WikiSQL span over a large number of tables and hence presents an unique challenge: the model must be able to not only generalize to new queries, but to new table schema. Finally, WikiSQL contains realistic data extracted from the web. This is evident in the distributionsofthenumberofcolumns,thelengthsofquestions,andthelengthofqueries,respectively shown in Figure 5. Another indicator of the variety of questions in the dataset is the distribution of question types, shown in Figure 4.

****.jsonl文件的格式：****

{

"phase":1,

"question":"who is the manufacturer for the order year 1998?",

"sql":{

"conds":[

[

0,

0,

"1998"

]

],

"sel":1,

"agg":0

},

"table\_id":"1-10007452-3"

}

****.tables.jsonl的格式：****

{

"id":"1-1000181-1",

"header":[

"State/territory",

"Text/background colour",

"Format",

"Current slogan",

"Current series",

"Notes"

],

"types":[

"text",

"text",

"text",

"text",

"text",

"text"

],

"rows":[

[

"Australian Capital Territory",

"blue/white",

"Yaa\u00b7nna",

"ACT \u00b7 CELEBRATION OF A CENTURY 2013",

"YIL\u00b700A",

"Slogan screenprinted on plate"

]

]

}

sel: 表中的哪一列被选中了，是这一列的索引值。

agg: 指的是aggregation operator的索引号，在lib/query.py中我们可以看到是这几个operator:

agg\_ops = ['', 'MAX', 'MIN', 'COUNT', 'SUM', 'AVG']

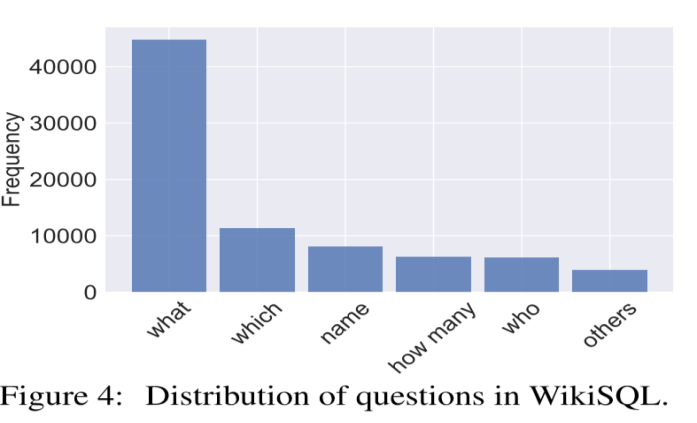
conds:三元组列表（column\_index，operator\_index，condition）where column = "value"

column\_index：列的索引

operator\_index：满足这个条件数字索引。也就是下面几种运算符：

cond\_ops = ['=', '>', '<', 'OP']

condition：条件的比较值，以字符串或浮点类型表示。也就是value





我们创建了数据集 Spider，它包含 200 个具备多个表的数据库、10181 个问题、5693 个对应的复杂 SQL 查询。所有这些由 11 名耶鲁大学学生标注完成，共耗时 1000 小时！

WikiSQL：SQL 查询和表的数量很多，但是所有 SQL 查询都很简单，仅包含 SELECT 和 WHERE 从句。此外，每个数据库都只是没有外键的简单的表。在 WikiSQL 上训练的模型在其它新数据库上仍然可以运行，但是该模型无法处理复杂的 SQL（如 GROUP BY、ORDER BY 或嵌套查询）和具备多个表和外键的数据库。

Spider 的范围最大，因此它是最复杂的跨域 text-to-SQL 数据集。为什么我们说它是最大的复杂跨域数据集呢？

大：超过 10000 个问题，6000 个对应的独特 SQL 查询。

复杂：大部分 SQL 查询覆盖几乎所有重要的 SQL 组件，包括 GROUP BY、ORDER BY、HAVING 和嵌套查询。此外，所有数据库都具备多个由外键链接的表。

跨域：包含 200 个复杂数据库。根据数据库类型，我们将 Spider 数据集分割成训练、开发和测试集。这样，我们就可以在未见过的数据库上测试系统性能。

****为什么大、复杂、跨域？****

首先，要训练一个深度学习模型，数据集越大，性能越好。其次，你当然希望训练数据尽可能多地覆盖更多场景，包括不同的 SQL 组件和数据库模式。这样，系统可以更好地适应多种情形。最后，为什么我们想要跨域数据？简单来讲，当你遇到新数据库时，你不想重新标注数据、重新训练一个新模型，这很浪费时间！

TableQA 中文标注数据库

4万条有标签数据作为训练集，1万条无标签数据作为测试集

{ "id":"a1b2c3d4", # 表格id "name":"Table\_a1b2c3d4", # 表格名称 "title":"表1：2019年新开工预测 ", # 表格标题 "header":[ # 表格所包含的列名 "300城市土地出让", "规划建筑面积(万㎡)", …… ], "types":[ # 表格列所相应的类型 "text", "real", …… ], "rows":[ # 表格每一行所存储的值 [ "2009年7月-2010年6月", 168212.4, …… ] ] }



针 对这个问题，目前有几个不同的解决思路：

陈程

我国的基于中文自然语言查询的数据库系统研究开始于上世纪末的八十至九十年代，主要有：

(I)

张亚南等人开发的

EAAD

系统模型

[4]

：研究涉及在多个实体以及中文实体上查询

ER

模型，联系、属性关系和相互依存的模型，并且给予一个正式的描述，目的是为了提高认识，扩展能力的便利性。

(2)

李保利等人开发的

WTCDIS

系统模型

[4]

：模型采用混合结构双层处理策略，以确定在查询中处于中心地位的关键短语，不需要输入用户使用完全标准化的查询语句。系统可处理单表及多表连接的相关简单查询，但无法通过选择性查询，或者使用了全部称谓的量词、复杂的嵌套查询，以及对条件进行分组等查询。

(3)

孟小峰、王珊等人开发的

NCHIQL

系统模型

[4]

回值，增强系统的纠错能力、实际使用能力和强壮能力。

(4)

崔宗军等人开发研究的

NLCQ

[4]

系统模型：结合中文自然语言语句表达的特点，实现了受约束的、基于中文自然语言的数据库查询系统。有词汇，语法切分分析和语义分析的阶段，并且在原有处理技术基础上提供了信息反馈的功能。

自然语言查询系统的研究开发工作在过去的近几年中取得了很大进展，但仍面临着这样那样的诸多问题：

(1)

自然语言查询系统的可再移植性的问题：一套自然语言处理系统，通用部分的可再用性往往不佳，而实际使用的系统又是面向实际应用的有限领域设计的，包括所使用的技术，如：数据结构模式匹配算法，语义语法分析分解等方面。使得系统在一个特定的区域内能够很好地运行工作，但难以迁移到该应用系统以外的其他领域。传统的多功能性自然语言处理模型与实用性存在着本质上无法解决的矛盾。

(2)

自然语言查询系统对于查询模糊性的处理问题：由于自然语言的特殊复杂性，一旦离开了相邻相关的知识库背景下，同样的语句将存在不同的含义，也有不同的组合之间的歧义。让数据库系统准确理解并分析句式句意所要表达的正确信息具有很大难度，所以要如何处理语言的不确定性一直困扰着自然语言技术的发展。

(3)

自然语言查询系统对于计算机如何理解省略部分的限制：人类历史悠久，拥有丰富的背景知识，大量的有效信息都可以通过推理获得，然而如何使一台计算机也拥有推理能力是非常值得思考的关键问题。

(4)

自然语言查询系统对于受限制的问题的理解程度：当下的查询系统一般仅限于对语句的理解，如果在已经限定好的范围内调整，过于狭窄，则理解语义覆盖面下降；若受限程度较弱，那么实现起来的难度就过大，查询系统的可用性同样难以得到有效提升。

：如何将人类自然语言的分析与数据库语言相结合成为了该系统模型研究的重点。在该类基础之上，提出了一组使用数据库的语义语言处理逻辑，以解决基于语义分割思路的算法。该算法的关键问题是如何使用最小匹配法匹配数据库中的数据，以及解析和数据库查询转换。只要有一个数据库查询的语义成分存在，你就可以得到一个返

本文开发设计的数据库自然语言查询接口吸取了其它模型的长处并摈弃其技术上的不足，使用了基于不确定有限自动机

(NFA)

模型进行自然语言处理。通过该接口，用户只需要输入中文自然语言查询语句，系统就会自动对输入的中文自然语言查询语句进行理解，包括词典分析、分词及语法等相结合的分析，将查询条件语句转换成为适当的中间语言，再实现中间语言向

SOL

的转换，从而执行对数据库的实际查询，最终得出所需的查询结果。

提取有关领域方面知识

实现让系统自动或半自动地提取领域知识，以建立一个专用的针对该领域的数据字典，这是实现系统的便携性的重要阶段。提取工艺技术被应用在了在许多

NLIDB

系统中。

领域知识的提取处理，对系统的可移植性具有很大影响，因为它的字典生成一个结果，它也将影响系统的可用性。提取方法的基本要求就是能够充分地降低系统用户的负担，并提高自动生成一个字典的能力。

领域知识比较重要。

为了提高查询准确率：一种是主动的解决方法，即优化加工分析，提高语言覆盖范围的方法；一种是被动的做法，即在不改变语言覆盖范围的前提条件下，用简单方便的方式提醒用户受约束的自然语言的范围，用以降低用户输入错误查询请求的几率。很明显的，第二种方法是简单、易于系统实现的，并且大大简化了系统的开发过程，但它并不是解决问题的根本办法。

数据库技术与系统的模型技术，特别是

ER

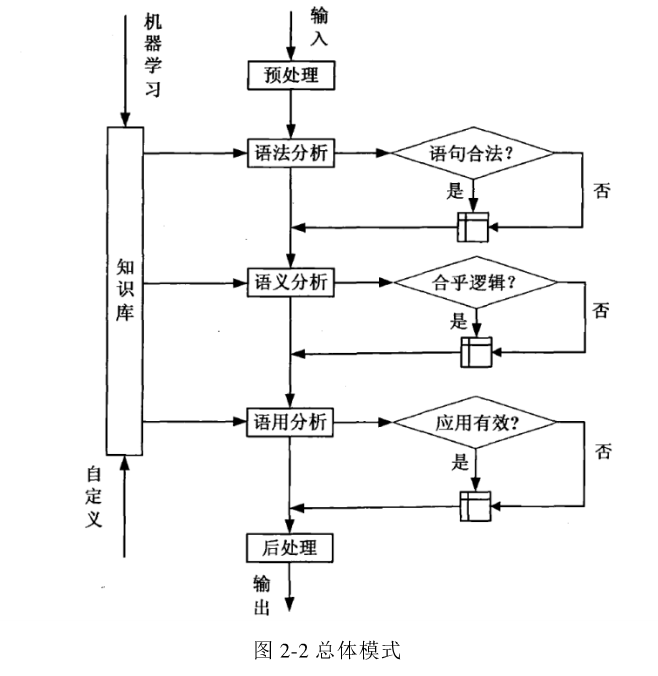
模型关系密切，而

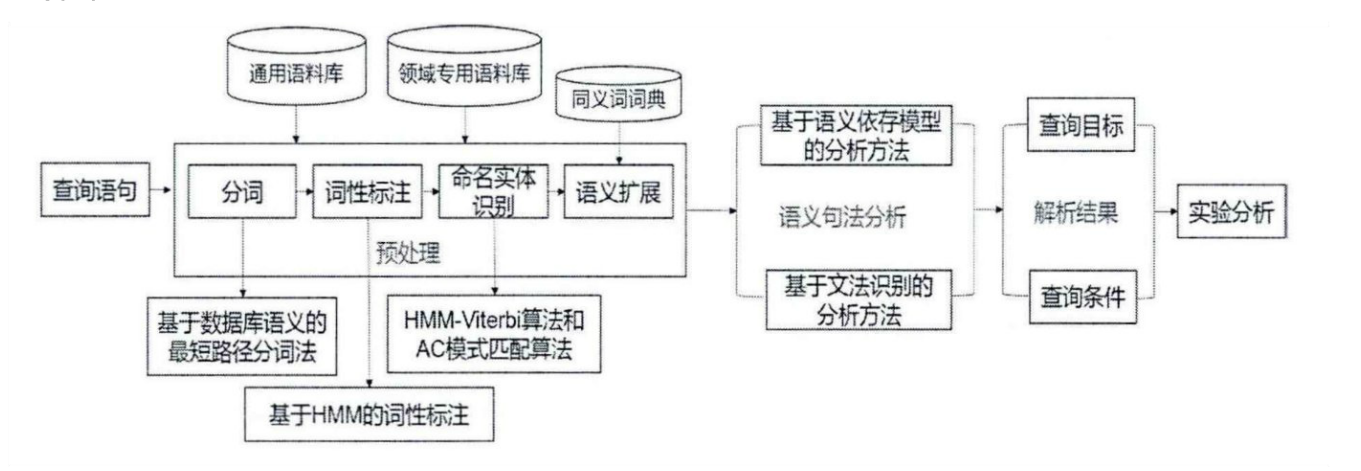
ER

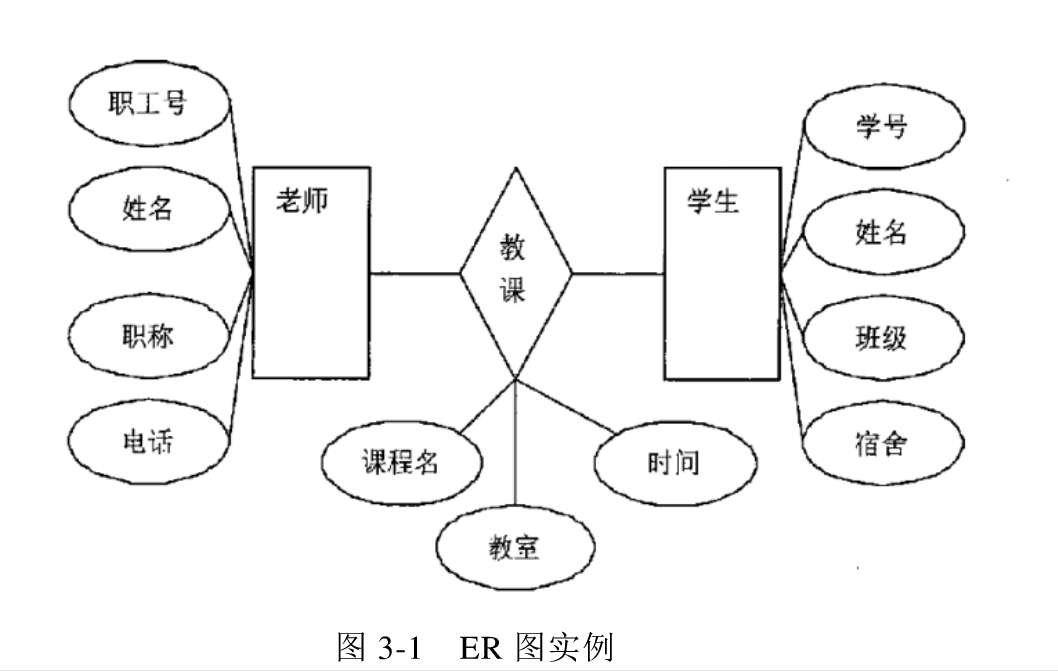
模型有可以便捷的转换为关系数据库。所以就能够通过使用

ER

模型，在数据库自然语言分析查询系统中，建立一座桥梁。







处理用户输入的查询语句的语法语义是该模块所要完成的主要功能。中间语言是经系统自动分词后形成的中文词组串，由中文自然语言查询文法自上向下检索，由左向右形成中文查询的中文词组串结构树，所有叶结点均对应实体

(

关联

)

或属性的语义所指。

知识库既是数据字典，分词知识库，专业领域字典

Jieba，最大匹配，基于规则

中间语言无需将用户输入的全部含义统统表示出来，只需要简单明了的把用户希望对数据库系统所要进行的查询请求表述明白即可。

（

2

）中间语言的设计需要迎合其桥梁作用连接的两个方面的表达需求（中

间语言以及数据库系统）。

（

3

）中间语言的设计应该保证其翻译出的语言是无歧义的，意义清楚明白，便于二次翻译成计算机语言。

（

4

）中间语言的设计是不可能在短期内达到最优的，需要开发人员不断的更新，再研究，再更新。所以就要求它一定要具有可重用性，便于二次开发。

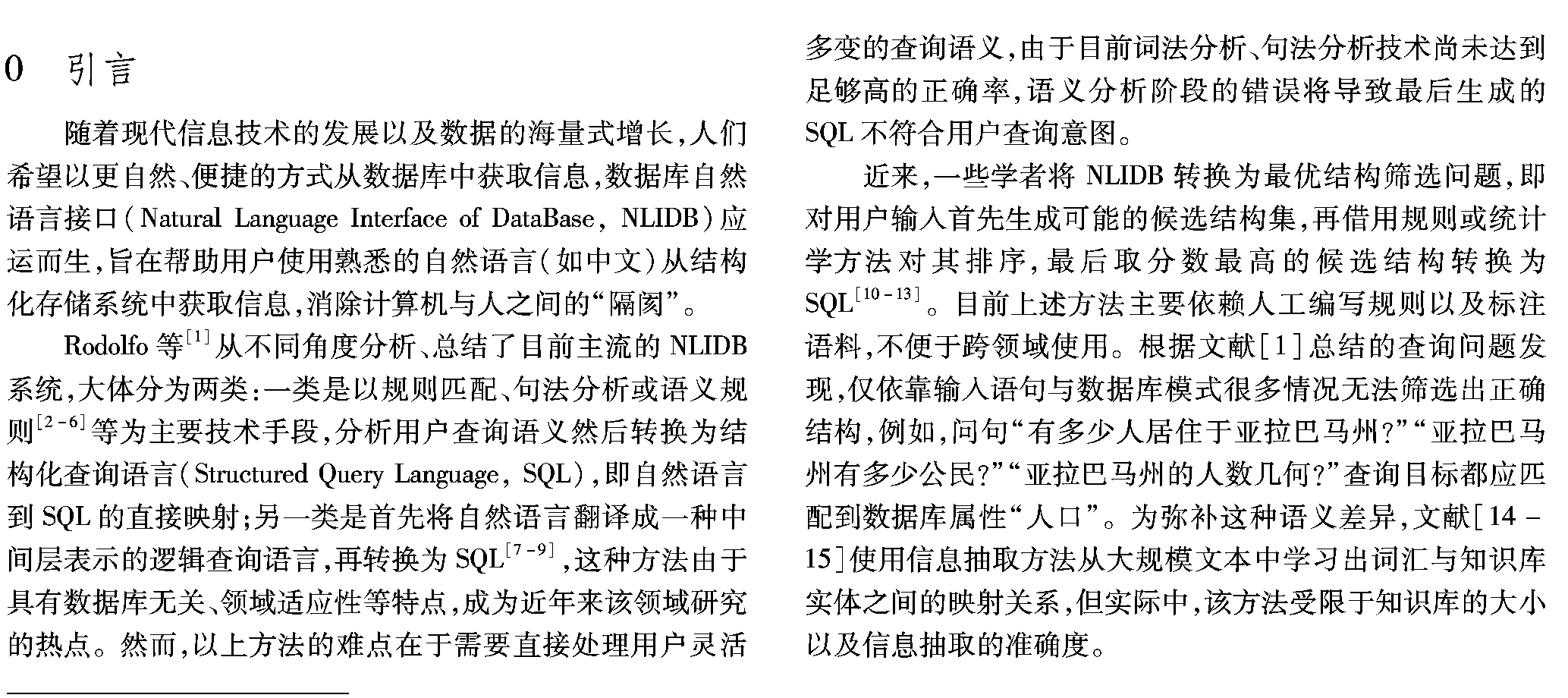
根据er模型对中间语言进行建模，最后翻译成sql

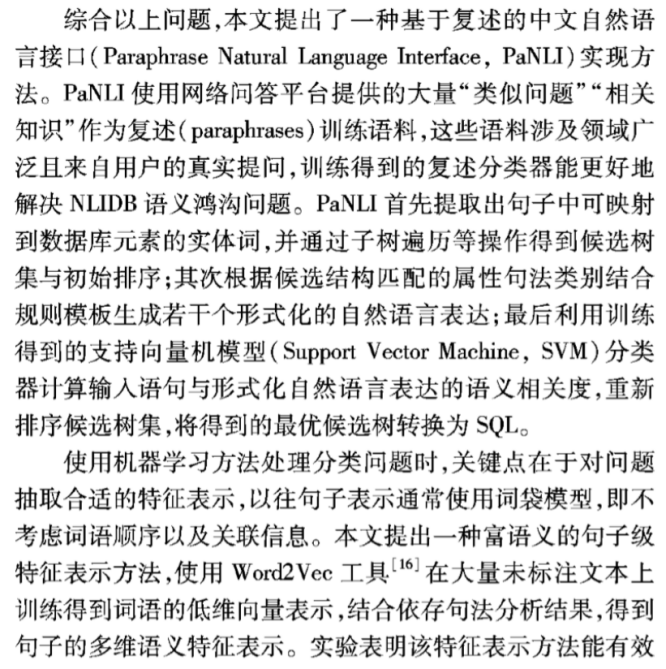
语料库设计，词法分析，语法分析。

以前没有数据集。

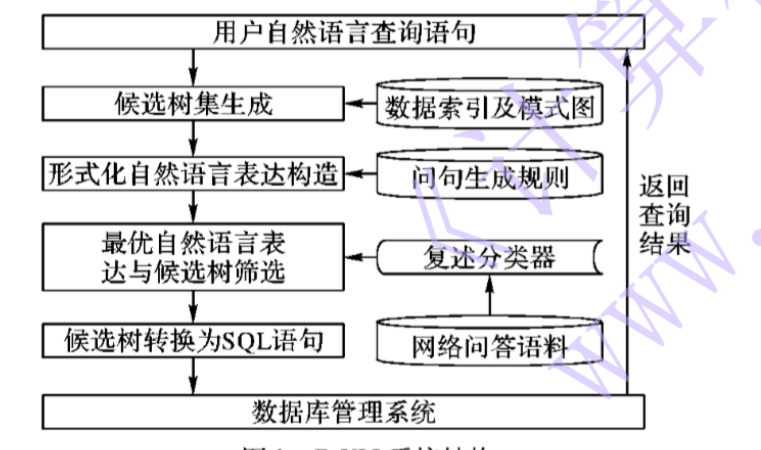
所以就建立语言知识，领域知识，数据库知识。主要是基于规则和统计。

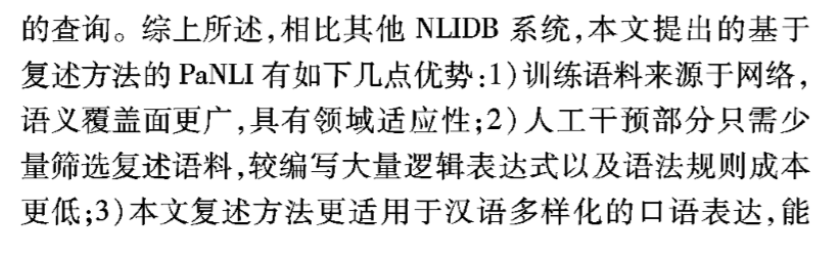
1. 传统算法

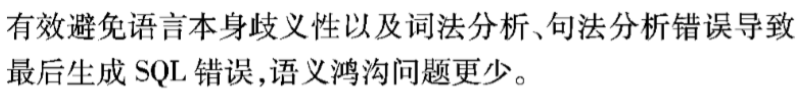


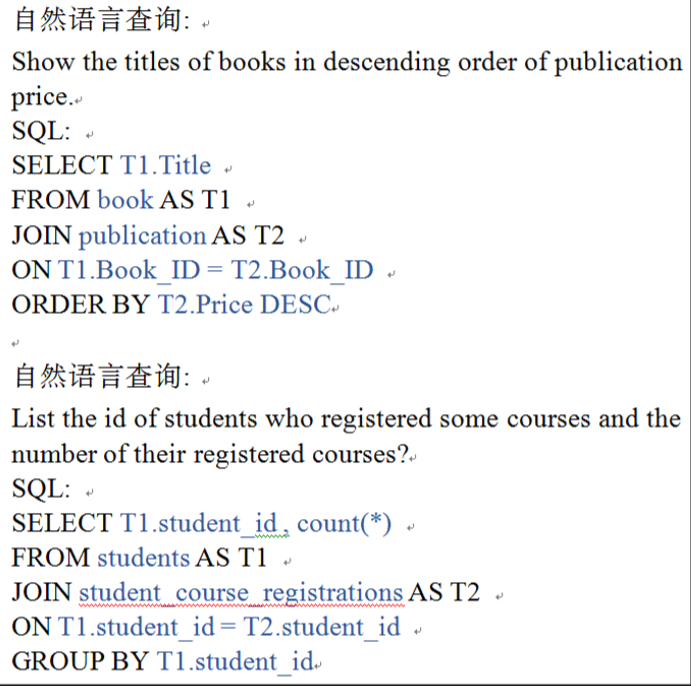


（ 1）流 水线方法：将自然语言查询转化为一种中间表达， 再将其映射为 SQL 语句；

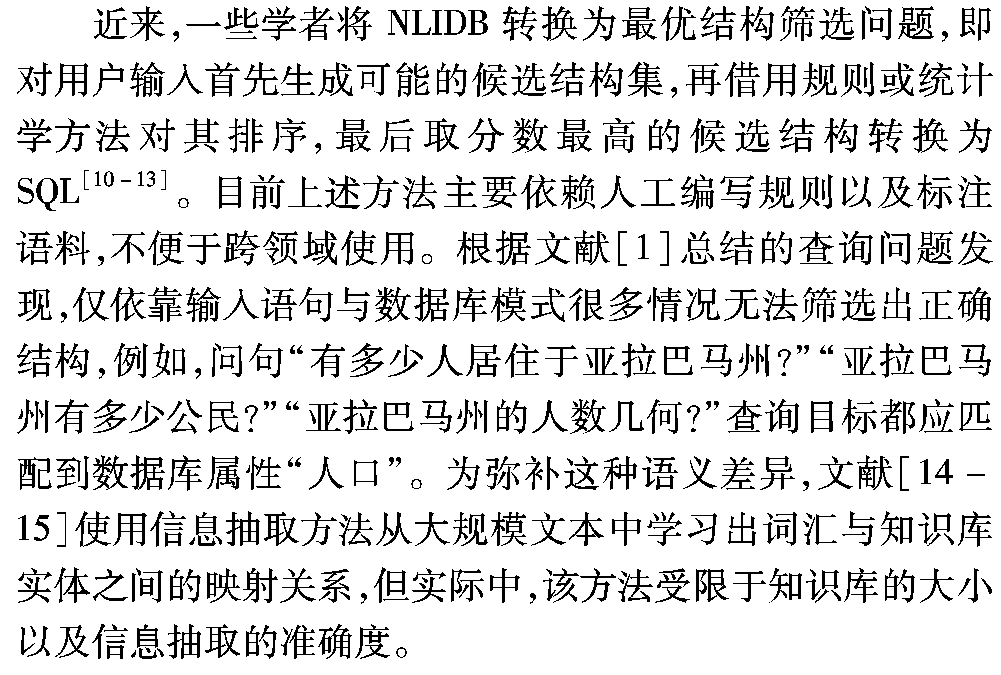


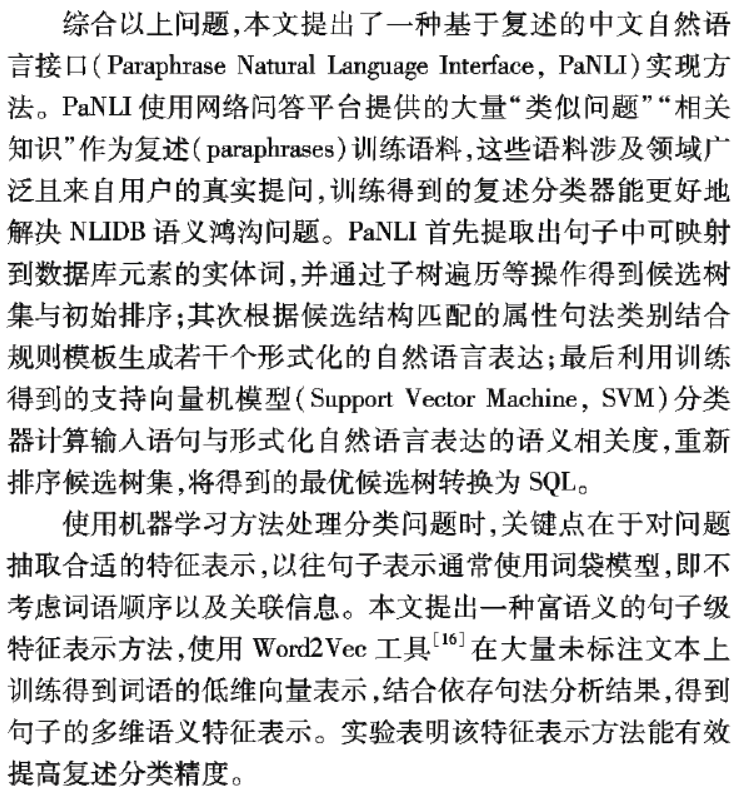






1 传统方法





1 候选集生成

2 形式化自然语言表达

3 最优自然语言表达与候选集筛选

4 候选树转换为sql语句

二

但是目前的主流方法是用深度学习方法去解决自然语言查询问题

Table2answer: Read the database and answer without SQL Tong Guo 1 Huilin Gao 2 1 Rokid AI Lab

Guo等人提出了一种解决方案，不用生成sql，直接去数据库的表中查找答案。

Semantic parsing is the task of mapping natural language to logic form. In question answering, semantic parsing can be used to map the question to logic form and execute the logic form to get the answer. One key problem for semantic parsing is the hard label work. We study this problem in another way: we do not use the logic form any more. Instead we only use the schema and answer info. We think that the logic form step can be injected into the deep model. The reason why we think removing the logic form step is possible is that human can do the task without explicit logic form. We use BERT-based model and do the experiment in the WikiSQL dataset, which is a large natural language to SQL dataset. Our experimental evaluations that show that our model can achieves the baseline results in WikiSQL dataset.

（2）深度学习方法：使用 端到端的神经网络来完成转化[1]。 流水线法将自然语言转化为中间表达的过 程依赖于自然语言查询的规则化描述，因此无法处 理一些复杂多变的自然语言描述。而近年来，随着 深度学习技术的不断収展，使用神经网络模型来处 理 NL2SQL 的工作也越収普遍。其优势较前者在于 不受自然语言描述多样性的限制，也能从复杂的表 达中提取出我们关注的语义信息。但现有 NL2SQL 的深度学习斱法存在着一定的局限性，需指定当前 自然语言查询对应数据库中的哪一张表，也就忽略了自然语言查询到对应查询表的映射，同时也无法 支持如图 1 中所示例的多表查询。

Seq2SQL，一种用于翻译语言语言问题的深层神经网络，用于对应ql查询。如图1所示，Seq2SQL由三个组件组成，它们利用SQL的结构来缩减生成查询的输出空间。此外，该算法利用基于策略的强化学习（RL）生成查询条件，由于其无序性，不适合用交叉熵损失进行优化。我们使用一个混合目标来训练Seq2SQL，将交叉熵损失和循环查询在数据库上执行的RL回报结合起来。这些特性允许Seq2SQL在查询生成方面获得最新的结果。接下来，我们将发布WikiSQL，一个包含80654个自然语言问题、SQL查询和SQL表的手工注释实例的语料库，这些实例是从Wikipedia的24241个HTML表中提取的。WikiSQL比以前提供逻辑形式和自然语言语句的语义分析数据集要大一个数量级。我们以原始JSON格式和SQL数据库的形式发布WikiSQL中使用的表。与WikiSQL一起，我们为用于循环查询执行的数据库发布了一个查询执行引擎，以学习策略。在WikiSQL上，Seq2SQL优于Dong以前最先进的语义解析模型

RELATED WORK Semantic Parsing. In semantic parsing for question answering (QA), natural language questions are parsed into logical forms that are then executed on a knowledge graph (Zelle & Mooney, 1996;

7

Wong & Mooney, 2007; Zettlemoyer & Collins, 2005; 2007). Other works in semantic parsing focus on learning parsers without relying on annotated logical forms by leveraging conversational logs (Artzi & Zettlemoyer, 2011), demonstrations (Artzi & Zettlemoyer, 2013), distant supervision(Cai&Yates,2013;Reddyetal.,2014),andquestion-answerpairs(Liangetal.,2011). Semantic parsing systems are typically constrained to a single schema and require hand-curated grammars to perform well2. Pasupat & Liang (2015) addresses the single-schema limitation by proposing the ﬂoating parser, which generalizes to unseen web tables on the WikiTableQuestions task. Our approach is similar in that it generalizes to new table schema. However, we do not require access to table content, conversion of table to an additional graph, nor hand-engineered features/grammar. Semanticparsingdatasets. Previoussemanticparsingsystemsweredesignedtoanswercomplex and compositional questions over closed-domain, ﬁxed-schema datasets such as GeoQuery (Tang & Mooney, 2001) and ATIS (Price, 1990). Researchers also investigated QA over subsets of largescale knowledge graphs such as DBPedia (Starc & Mladenic, 2017) and Freebase (Cai & Yates, 2013; Berant et al., 2013). The dataset “Overnight” (Wang et al., 2015) uses a similar crowdsourcing process to build a dataset of natural language question, logical form pairs, but has only 8 domains. WikiTableQuestions (Pasupat&Liang,2015)isacollectionofquestionandanswers,also overalargequantityoftablesextractedfromWikipedia. However,itdoesnotprovidelogicalforms whereas WikiSQL does. In addition, WikiSQL focuses on generating SQL queries for questions over relational database tables and only uses table content during evaluation. Representationlearningforsequencegeneration. Dong&Lapata(2016)’sattentionalsequence to sequence neural semantic parser, which we use as the baseline, achieves state-of-the-art results on a variety of semantic parsing datasets despite not utilizing hand-engineered grammar. Unlike their model, Seq2SQL uses pointer based generation akin to Vinyals et al. (2015) to achieve higher performance, especially in generating queries with rare words and column names. Pointer models have also been successfully applied to tasks such as language modeling (Merity et al., 2017), summarization (Gu et al., 2016), combinatorial optimization (Bello et al., 2017), and question answering (Seo et al., 2017; Xiong et al., 2017). Other interesting neural semantic parsing models are the Neural Programmer (Neelakantan et al., 2017) and the Neural Enquirer (Yin et al., 2016). Mou et al. (2017) proposed a distributed neural executor based on the Neural Enquirer, which efﬁciently executes queries and incorporates execution rewards in reinforcement learning. Our approach is different in that we do not access table content, which may be unavailable due to privacy concerns. We also do not hand-engineer model architecture for query execution and instead leverage existing database engines to perform efﬁcient query execution. Furthermore, in contrast to Dong & Lapata (2016) and Neelakantan et al. (2017), we use policy-based RL in a fashion similar to Liang et al. (2017), Mou et al. (2017), and Guu et al. (2017), which helps Seq2SQL achieve state-of-the-art performance. Unlike Mou et al. (2017) and Yin et al. (2016), we generalize across natural language questions and table schemas instead of across synthetic questions on a single schema. Naturallanguageinterfacefordatabases. Oneprominentworksinnaturallanguageinterfacesis PRECISE(Popescuetal.,2003),whichtranslatesquestionstoSQLqueriesandidentiﬁesquestions that it is not conﬁdent about. Giordani & Moschitti (2012) translate questions to SQL by ﬁrst generating candidate queries from a grammar then ranking them using tree kernels. Both of these approaches rely on high quality grammar and are not suitable for tasks that require generalization to new schema. Iyer et al. (2017) also translate to SQL, but with a Seq2Seq model that is further improved with human feedback. Seq2SQL outperforms Seq2Seq and uses reinforcement learning instead of human feedback during training.

sqlnet

X-sql

Spider