Tao Yu Follow
Sep 26, 2018 · 6 min read · Listen

Spider: One More Step Towards Natural Language Interfaces to Databases

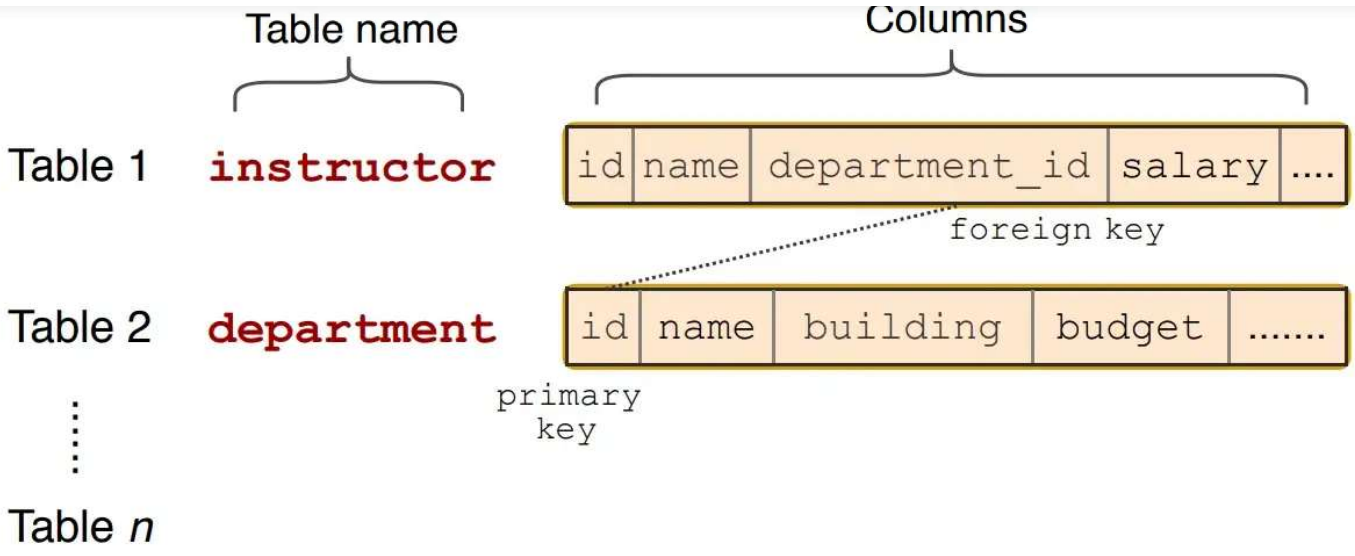
Spider: Yale Semantic Parsing and Text-to-SQL Challenge

Yale Spider is a large dataset for complex and cross-domain semantic parsing and text-to-SQL Task introduced by our...

yale-lily.github.io

Introduction

The amount of data produced daily has been increasing exponentially since the start of the new millennia. Most of this data is stored in relational databases. In the past, access to this data has been the interest of mostly large companies, who are able to query the data using structured query languages (SQL). With the growth of mobile phones, more and more personal data is being stored. Thus, more and more people from different backgrounds are trying to query and use their own data. Despite the meteoric rise in the popularity of data science, most people do not have adequate knowledge to write SQL and query their data. Moreover, most people do not have time to learn and understand SQL. Even for SQL experts, writing similar queries, again and again, is a tedious task. Due to this fact, the vast amount of data available today cannot be effectively accessed.



Annotators create:

Complex question

What are the name and budget of the departments with average instructor salary greater than the overall average?

Complex SQL

```
SELECT T2.name, T2.budget
FROM instructor as T1 JOIN department as
T2 ON T1.department_id = T2.id
GROUP BY T1.department_id
HAVING avg(T1.salary) >
    (SELECT avg(salary) FROM instructor)
```

An Example of Annotated Question and SQL Pair

If you do not understand the long piece of SQL code on the left, do not worry! This is where natural language interfaces to databases come in. The goal is to allow you to talk to your data directly using human language! Thus, these interfaces help users of any background easily query and analyze a vast amount of data.

How to Build this Interface?

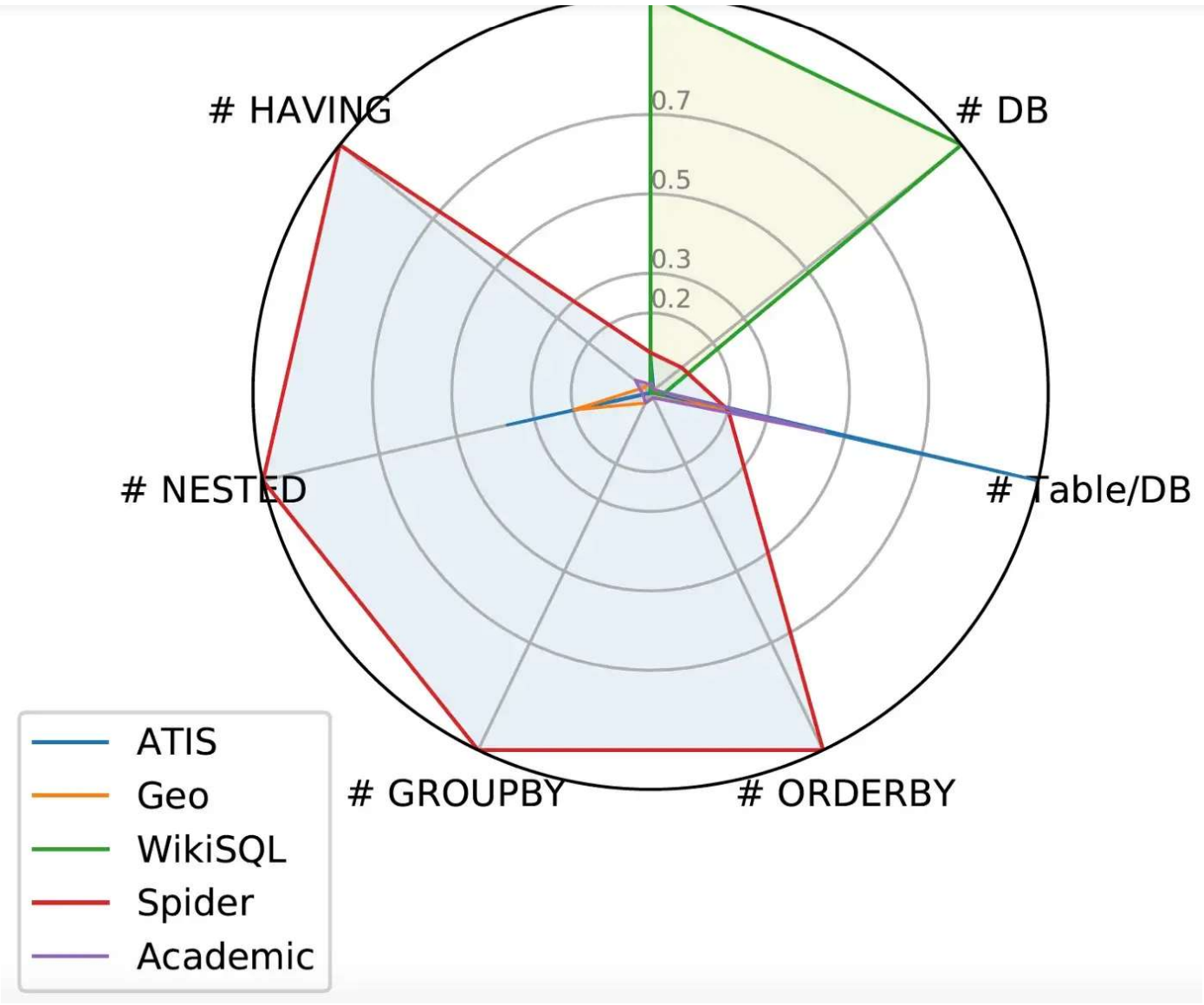
To build this kind of natural language interface, the system has to understand users'

Good Data is Scarce!

However, one critical problem comes up: where can we find a large amount of question and SQL pair labels? Creating this kind of dataset is very time-consuming because annotators have to understand the database schema, ask questions and then write SQL answers, all of which requires very specific database knowledge. One thing which makes this even more difficult is that the number of non-private databases with multiple tables are very limited. To address the need for a large and high-quality dataset for this task, we are pleased to introduce ***Spider***, which consists of 200 databases with multiple tables, 10,181 questions, and 5,693 corresponding complex SQL queries. All of them are written by 11 Yale students, spending a total of 1,000 man-hours!

Why Spider?

Even though creating such data is hard, there exist some similar kinds of data resources such as SQL queries in the traditional 9 single databases, including ATIS, GeoQuery, Scholar, Advising etc. and WikiSQL. So, why you should choose Spider? Let's take a look at the following chart:

Spider Chart of Some Text-to-SQL Datasets

- *ATIS, Geo, Academic*: Each of these datasets contains only a single database. Most of them contain only < 500 unique SQL queries. Basically, models trained on these datasets can work only for the specific database. They would totally fail once you switch database domains.

- *WikiSQL*: The numbers of SQL queries and tables are significantly large, but all SQL queries are simple, which only cover SELECT and WHERE clauses. Also, each database is only a simple table without any foreign key. Models trained on WikiSQL still work when tested on a different new database. However, the model cannot

Spider spans the largest area in the chart, making it the first complex and cross-domain text-to-SQL dataset! Why do we call it a large complex and cross-domain dataset?

- *Large:* Over 10,000 questions, and about 6,000 corresponding, unique SQL queries
- *Complex*: Most of the SQL queries cover almost all important SQL components including GROUP BY, ORDER BY, HAVING and nested queries. Also, all databases have multiple tables linked by some foreign keys.
- *Cross-domain*: consists of 200 complex databases. We split Spider into train, dev, and test based on databases. In this way, we can test the systems' performances on unseen databases.

Why Large, Complex, and Cross-Domain?

First, for training a deep learning model, basically, the larger the dataset, the better the performance. Second, you want your training data to cover as many scenarios as possible, including different SQL components and database schema. In this way, your system can learn from them so that the system does not fail in most cases. Finally, why we should care about cross-domain data? Simply, you do not want to relabel data and retrain a new model when you get a new database. This wastes a lot of time!

Excited? Download Spider!

taoyds/spider

scripts and baselines for Spider: Yale complex and cross-domain semantic parsing and text-to-SQL challenge ...

github.com

You can find the Spider dataset and leaderboard at *our project page* or *Github page*! We


Other Challenges?

Once we have a nice dataset, what are other challenges we have to solve in order to build real-world natural language interfaces to databases? From the perspective of natural language processing (NLP), there are three main tasks:

- **Natural language understanding:** The system has to understand users' questions, which could be ambiguous, random and diverse.
- **Database schema representation:** Database can be very complex, with over hundreds of columns, many tables, and foreign keys.
- **Complex SQL decoding/generation:** Once the system understands the user's question and the database's schema to which the user is querying, it has to generate the corresponding SQL answer. However, SQL queries can be very complex and include nested queries with multiple conditions.

Citation Credit

When you use the Spider dataset, we would appreciate it if you cite the following:

```
@inproceedings{Yu&al.18c,
  year =         2018,
  title =        {Spider: A Large-Scale Human-Labeled Dataset for
Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task},
  booktitle =    {EMNLP},
  author =       {Tao Yu and Rui Zhang and Kai Yang and Michihiro
Yasunaga and Dongxu Wang and Zifan Li and James Ma and Irene Li and
Qingning Yao and Shanelle Roman and Zilin Zhang and Dragomir Radev }
}
```

A List of Some Related Work

This field has been studied for decades mainly from both NLP and Database communities. Here is a short list of recent related work (listed by publish year):

SyntaxSQL-Net: Syntax Tree Networks for Complex and Cross-Domain Text-to-SQL

- *Zero-shot Parser:* Decoupling Structure and Lexicon for Zero-Shot Semantic Parsing
- *Coarse2fine*: Coarse-to-Fine Decoding for Neural Semantic Parsing
- *SQL Evaluation Methodology*: Improving Text-to-SQL Evaluation Methodology
- *User Feedback via Dialog:* DialSQL: Dialogue Based Structured Query Generation
- *Involving Context in the Task:* Learning to Map Context-Dependent Sentences to Executable Formal Queries
- *TypeSQL:* TypeSQL: Knowledge-based Type-Aware Neural Text-to-SQL Generation
- *SQLNet:* SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning
- *Seq2SQL:* Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning
- *Syntactic Neural Models:* A Syntactic Neural Model for General-Purpose Code Generation, Abstract Syntax Networks for Code Generation and Semantic Parsing
- *NL2SQL:* Learning a Neural Semantic Parser from User Feedback
- *Seq2Tree:* Learning a Neural Semantic Parser from User Feedback
- *NaLIR*: Constructing an Interactive Natural Language Interface for Relational Databases

Also, other related talks, blogs, or books:

- How to Talk to Your Database
- ACL 2018 Tutorial on Neural Semantic Parsing
- Natural Language Data Management and Interfaces


Finally, some work in other related fields:

- Cross-domain Semantic Parsing via Paraphrasing
- Building Natural Language Interfaces to Web APIs
- On Generating Characteristic-rich Question Sets for QA Evaluation