
Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task

Spider is a large human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task (natural language interfaces for relational databases). It is released along with our EMNLP 2018 paper: Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. This repo contains all code for evaluation, preprocessing, and all baselines used in our paper. Please refer to the task site for more general introduction and the leaderboard.

:+1: 03/20/2022: We open-sourced a simple but SOTA model (just T5) for the task! Please check out our code in the UnifiedSKG repo!!

Changelog

-11/15/2020 We will use Test Suite Accuracy as our official evaluation metric for Spider, SParC, and CoSQL. Please find the evaluation code from here. - 08/03/2020 Corrected `column_name` and `column_name_original` mismatches in 2 dbs (`scholar` and `formula_1`) in `tables.json`, and reparsed SQL queries (this only affects some models (e.g. RATSQ) which use our parsed SQL as the SQL input). Please download the Spider dataset from the page again. - 06/07/2020 We corrected some annotation errors and label mismatches (not errors) in Spider dev and test sets (~4% of dev examples updated, click here for more details). Please download the Spider dataset from the page again. - 01/16/2020 For value prediction (in order to compute the execution accuracy), your model should be able to 1) copy from the question inputs, 2) retrieve from the database content (database content is available), or 3) generate numbers (e.g. 3 in "LIMIT 3"). - 1/14/2019 The submission tutorial is ready! Please follow it to get your results on the unreleased test data. - 12/17/2018 We updated 7 sqlite database files. Please download the Spider data from the official website again. Please refer to the issue 14 for more details. - 10/25/2018: evaluation script is updated so that the table in `count(*)` cases will be evaluated as well. Please check out the issue 5 for more info. Results of all baselines and syntaxSQL on the papers are updated as well. - 10/25/2018: to get the latest SQL parsing results (a few small bugs fixed), please use `preprocess/parse_raw_json.py` to update. Please refer to the issue 3 for more details.

Citation

The dataset is annotated by 11 college students. When you use the Spider dataset, we would appreciate it if you cite the following:

```
1 @inproceedings{Yu&al.18c,  
2   title      = {Spider: A Large-Scale Human-Labeled Dataset for Complex  
   and Cross-Domain Semantic Parsing and Text-to-SQL Task},  
3   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  
   }  
4   booktitle  = "Proceedings of the 2018 Conference on Empirical Methods  
   in Natural Language Processing",  
5   address    = "Brussels, Belgium",  
6   publisher  = "Association for Computational Linguistics",  
7   year       = 2018  
8 }
```

Installation

`evaluation.py` and `process_sql.py` are written in Python 3. Enviroment setup for each base-line is in README under each baseline directory.

Data Content and Format

Question, SQL, and Parsed SQL Each file `intrain.json` and `dev.json` contains the following fields: - `question`: the natural language question - `question_toks`: the natural language question tokens - `db_id`: the database id to which this question is addressed. - `query`: the SQL query corresponding to the question. - `query_toks`: the SQL query tokens corresponding to the question. - `sql`: parsed results of this SQL query using `process_sql.py`. Please refer to `parsed_sql_examples.sql` in the `preprocess` directory for the detailed documentation.

```
1  {  
2      "db_id": "world_1",  
3      "query": "SELECT avg(LifeExpectancy) FROM country WHERE Name  
   NOT IN (SELECT T1.Name FROM country AS T1 JOIN  
   countrylanguage AS T2 ON T1.Code = T2.CountryCode WHERE T2  
   .Language = \"English\" AND T2.IsOfficial = \"T\")",  
4      "query_toks": ["SELECT", "avg", "(", "LifeExpectancy", ")", "  
   FROM", "...],  
5      "question": "What is average life expectancy in the countries  
   where English is not the official language?",  
6      "question_toks": ["What", "is", "average", "life", "...],  
7      "sql": {  
8          "except": null,  
9          "from": {  
10             "conds": [],  
11             "table_units": [  
12                 ...
```

```

13     },
14     "groupBy": [],
15     "having": [],
16     "intersect": null,
17     "limit": null,
18     "orderBy": [],
19     "select": [
20         ...
21     ],
22     "union": null,
23     "where": [
24         [
25             true,
26             ...
27             {
28                 "except": null,
29                 "from": {
30                     "conds": [
31                         [
32                             false,
33                             2,
34                             [
35                                 ...
36                             ],
37                             "groupBy": [],
38                             "having": [],
39                             "intersect": null,
40                             "limit": null,
41                             "orderBy": [],
42                             "select": [
43                                 false,
44                                 ...
45                             ],
46                             "union": null,
47                             "where": [
48                                 [
49                                     false,
50                                     2,
51                                     [
52                                         0,
53                                         ...
54                                     ]
55                                 ]
56                             ]
57                         ]
58                     ]
59                 }
60             ]
61         ]
62     ]
63 }

```

Tables `tables.json` contains the following information for each database: - `db_id`: database id - `table_names_original`: original table names stored in the database. - `table_names`: cleaned and normalized table names. We make sure the table names are meaningful. [to be changed] - `column_names_original`: original column names stored in the database. Each column looks like: `[0, "id"]`. 0 is the index of table names in `table_names`, which is `city` in this case.

"id" is the column name. - `column_names`: cleaned and normalized column names. We make sure the column names are meaningful. [to be changed] - `column_types`: data type of each column - `foreign_keys`: foreign keys in the database. [3, 8] means column indices in the `column_names`. These two columns are foreign keys of two different tables. - `primary_keys`: primary keys in the database. Each number is the index of `column_names`.

```
1  {
2      "column_names": [
3          [
4              0,
5              "id"
6          ],
7          [
8              0,
9              "name"
10         ],
11         [
12             0,
13             "country code"
14         ],
15         [
16             0,
17             "district"
18         ],
19         .
20         .
21         .
22     ],
23     "column_names_original": [
24         [
25             0,
26             "ID"
27         ],
28         [
29             0,
30             "Name"
31         ],
32         [
33             0,
34             "CountryCode"
35         ],
36         [
37             0,
38             "District"
39         ],
40         .
41         .
42         .
43     ],
44     "column_types": [
```

```

45     "number",
46     "text",
47     "text",
48     "text",
49     .
50     .
51     .
52 ],
53 "db_id": "world_1",
54 "foreign_keys": [
55     [
56         3,
57         8
58     ],
59     [
60         23,
61         8
62     ]
63 ],
64 "primary_keys": [
65     1,
66     8,
67     23
68 ],
69 "table_names": [
70     "city",
71     "sqlite_sequence",
72     "country",
73     "country language"
74 ],
75 "table_names_original": [
76     "city",
77     "sqlite_sequence",
78     "country",
79     "countrylanguage"
80 ]
81 }

```

Databases All table contents are contained in corresponding SQLite3 database files.

Evaluation

Update 11/15/20: We will use Test Suite Accuracy as our official evaluation metric for Spider, SPaRC, and CoSQL. Please find the evaluation code from [here](#). Our evaluation metrics include Component Matching, Exact Matching, and Execution Accuracy. For component and exact matching evaluation, instead of simply conducting string comparison between the predicted and gold SQL queries, we decompose each SQL into several clauses, and conduct set comparison in each SQL clause.

For Execution Accuracy, our current models do not predict any value in SQL conditions so that we do not provide execution accuracies. However, we encourage you to provide it in the future submissions. For value prediction, you can assume that a list of gold values for each question is given. Your model has to fill them into the right slots in the SQL.

Please refer to our paper and this page for more details and examples.

```
1 python evaluation.py --gold [gold file] --pred [predicted file] --etype  
  [evaluation type] --db [database dir] --table [table file]  
2  
3 arguments:  
4   [gold file]           gold.sql file where each line is `a gold SQL \t  
   db_id`  
5   [predicted file]      predicted sql file where each line is a predicted  
   SQL  
6   [evaluation type]     "match" for exact set matching score, "exec" for  
   execution score, and "all" for both  
7   [database dir]        directory which contains sub-directories where  
   each SQLite3 database is stored  
8   [table file]          table.json file which includes foreign key info of  
   each database
```

FAQ