Leveraging Adjective-Noun Phrasing Knowledge for Comparison Relation Prediction in Text-to-SQL

Haoyan Liu[1]*, **Lei Fang**[2], **Qian Liu**[3], **Bei Chen**[2], **Jian-Guang Lou**[2], **Zhoujun Li**[1]
[1]State Key Lab of Software Development Environment, Beihang University, China
[2]Microsoft Research, Beijing, China
[3]State Key Lab of Virtual Reality Technology and Systems, Beihang University, China
{haoyan.liu, qian.liu, lizj}@buaa.edu.cn; {leifa, beichen, jlou}@microsoft.com

Abstract

One key component in text-to-SQL is to predict the comparison relations between columns and their values. To the best of our knowledge, no existing models explicitly introduce external common knowledge to address this problem, thus their capabilities of predicting comparison relations are limited beyond training data. In this paper, we propose to leverage adjective-noun phrasing knowledge mined from the web to predict the comparison relations in text-to-SQL. Experimental results on both the original and the re-split Spider dataset show that our approach achieves significant improvement over state-of-the-art methods on comparison relation prediction.

1 Introduction

Text-to-SQL (Yaghmazadeh et al., 2017; Zhong et al., 2017), which aims at mapping natural language to SQL queries, is one of the most important tasks in natural language processing. Most state-of-the-art models are end-to-end neural network based models (Zhang et al., 2017; Xu et al., 2017; Yu et al., 2018a; Herzig and Berant, 2018; Dong and Lapata, 2018; Yu et al., 2018b), which mainly extend the Seq2Seq architecture with some complicated network structures. As shown by Yu et al. (2018b), the performances of most methods are overstated, because they just match semantic parsing results, rather than truly understand the meanings of inputs (Finegan-Dollak et al., 2018). In this paper, we study the comparison relation prediction problem, as the comparison relations between columns and their values are not well understood by existing methods.

In SQL queries, comparison relations are expressed by the comparison operators ($=, \neq, <, \leq, >, \geq$) and value ordering keywords (`ASC`, `DESC` in `ORDER BY` expression). In most text-to-SQL models, the comparison relations are either generated using Seq2Seq architectures (Zhong et al., 2017; Dong and Lapata, 2018), or predicted using classifiers trained with output decoding features (Xu et al., 2017; Yu et al., 2018b). We give an example to show that external common knowledge is indispensable to truly understand the comparison relations on unseen data.

rank	name	time	birth_date
1	Martina	11′9	19800930
2	Mirjana	12′5	19820309
3	Justine	12′9	19820601

Table 1: Player Basic Information.

Table 1 shows the basic information about the athletes of the 100-meter sprint. Given the query *"what is the name of the oldest player ?"*, the goal is to generate the SQL "`SELECT name FROM players ORDER BY birth_date ASC LIMIT 1`". It should be noted that `ASC` represents the common knowledge that small value in `birth_date` column means *"old"*. Supposing that there are columns named `age` and some queries containing *"old"* in the training data, it's easy for the trained models to remember that *"old"* means selecting a large value from column `age`. But if `birth_date` is unseen in the training data, there is little chance to predict the comparison relation correctly without the common knowledge that *"age"* and *"birth_date"* both represent age but have opposite value polarities. Similarly, for query *"fast runner"*, models should select a large value from column `speed` or a small value from column `time`. There are some related works that address the intensity of adjectives (De Melo and Bansal, 2013; Ruppenhofer et al., 2014; Sharma et al., 2017), however,

*This work was done when the first author was an intern at Microsoft Research Asia.

no existing work studies the relations between the value polarities of adjective-noun phrasing pairs.

In this paper, we propose to explicitly incorporate the column value polarities as external knowledge in text-to-SQL models. Our goal is to scale the capabilities of existing models on comparison relation prediction to unseen data. The column value polarities are named as "adjective-noun phrasing knowledge", which can be easily constructed from the web. We further formulate the phrasing knowledge as feature vectors, which can be generically fed to existing models. Experimental results on both the original and the re-split Spider dataset show that our approach achieves significant improvement over state-of-the-art methods on comparison relation prediction.

2 Adjective-Noun Phrasing Knowledge

There are 3 steps to construct the knowledge: 1) adjective-noun pair candidate extraction, 2) value polarity mining, and 3) adjective-noun pair clustering. The adjective-noun phrasing knowledge will be two clusters, each of which is a list of adjective-noun pairs with the same value polarity.

2.1 Adjective-Noun Pair Candidate Extraction

As the value polarities depend on both adjectives and nouns, we first extract adjective-noun pairs that could be candidates of the phrasing knowledge. Typically, table column names could be considered as hypernyms of the corresponding cell values. Motivated by this, we gather a list of noun candidates, which consists of the concepts (hypernyms) from Microsoft Concept Graph[1] (Cheng et al., 2015; Wu et al., 2012) and the column names whose value types are number or date in the Web Table Corpora[2] (Lehmberg et al., 2016).

To extract the adjectives that modify the noun candidates, we introduce two POS tag patterns:

```
(1) [ADJ] [NOUN]
(2) [NOUN] is|are|was|were|be [ADJ]
```

where `[ADJ]` and `[NOUN]` are the POS tags for adjectives and nouns, respectively. We apply these two patterns to Wikipedia dump[3] to extract adjective-noun pair candidates. Taking *price* as an example, we would obtain the adjectives: *high*, *expensive*, and *cheap*, etc.

[1] https://concept.research.microsoft.com

[2] http://webdatacommons.org/webtables/

[3] https://dumps.wikimedia.org/enwiki/

2.2 Value Polarity Mining

As shown in Table 1, the value polarities of `age` and `birth_date` are opposite though both words are about *"age"*. We propose to mine the value polarities from the Web Table Corpora automatically. We assume that for two columns in the same table, if their corresponding values have negative correlations, they have opposite value polarities[4]. We use the Spearman's ρ coefficients to measure the correlations between two columns. For each two columns with value type number or date, if the frequency of their co-occurrences is above 20 and the coefficients ρ are below -0.9 for more than 50% co-occurrences in the corpora, the two columns are determined to have opposite value polarities.

2.3 Adjective-Noun Pair Clustering

So far, we have obtained the adjective-noun pairs and value polarity relations for nouns. However, it is still unclear whether the polarity is positive or negative. Positive (negative) polarity means that large (small) column values shall be selected when the noun is modified by an adjective. For example, the polarities of <`age`, *old*> and <`price`, *expensive*> are positive, while <`age`, *young*> and <`price`, *cheap*> are negative. Our goal is to separate the adjective-noun pairs into two clusters based on the value polarities.

Supposing that we know the polarity of <`age`, *old*> is positive, we can derive that the polarities of <`age`, *young*> and <`birth_date`, *old*> are negative, because *"old"* and *"young"* are antonyms, and the value polarities of `age` and `birth_date` are opposite. Motivated by this, we extend and group the adjective-noun pairs into clusters based on the following rules:

- the polarities of two pairs are the same if the adjectives are synonyms and the nouns are the same, or the nouns are synonyms and the adjectives are the same;
- the polarities of two pairs are the opposite if the adjectives are antonyms and the nouns are the same, or the nouns have opposite value polarities and the adjectives are the same.

It should be noted that each cluster contains two sets of pairs with opposite polarities. We manually assign polarity labels to clusters in top sizes or

[4] We also studied the assumption that columns with positive correlations have the same value polarities, but find that it introduces much noise.

with high total frequencies. One potential issue is that there might be conflicts due to the synonyms and antonyms resources, i.e., there may be two pairs assigned opposite polarities in the same cluster. In practice, there are only a few such cases, and we separate them into individual clusters and manually label these pairs. After that, we obtain the adjective-noun phrasing knowledge organized in two clusters, each of which is a list of adjective-noun pairs with the same polarity.

3 Phrasing Knowledge As Features

We propose to formulate knowledge as features to incorporate the adjective-noun phrasing knowledge. It is very generic, because knowledge feature can be easily combined with the input or hidden output of existing neural models.

3.1 Baseline Models

We use Spider (Yu et al., 2018c) as the dataset. We do not use other datasets like WikiSQL (Zhong et al., 2017) because WikiSQL queries that contain comparison relations are very simple, and do not require additional knowledge for understanding. We introduce the knowledge feature to two baseline models, namely syntaxSQLNet[5] (Yu et al., 2018b) and SQLNet[6] (Xu et al., 2017; Yu et al., 2018c). We do not use other syntax tree-based or sequence-based baselines like coarse-to-fine (Dong and Lapata, 2018) because they cannot handle the Spider dataset.

Currently[7], syntaxSQLNet achieves the best performance on Spider. SyntaxSQLNet and SQLNet solve the text-to-SQL task using a sequence-to-set structure. They decompose the SQL generation procedure into multiple individual modules. The comparison relation prediction consists of two parts: the comparison operator prediction in `OP` module of `WHERE` and `GROUP Having` components and ordering prediction in `DESC/ASC/LIMIT(DAL)` module of `ORDER BY` component. Both the two modules are classifiers defined as:

$$p(y) = \mathcal{F}(V \tanh(WX)), \qquad (1)$$

where X represents the input features; y is the output label; and W and V are trainable parameters.

[5] https://github.com/taoyds/syntaxSQL

[6] https://github.com/taoyds/spider/tree/master/baselines/sqlnet

[7] As of Mid-May when we prepare this submission.

Function $\mathcal{F}$ is defined as `sigmoid` in SQLNet and `softmax` in syntaxSQLNet. The `OP` module predicts the comparison operator $y \in \{=, >, <, >=, <=, !=, \texttt{LIKE}, \texttt{NOTIN}, \texttt{IN}, \texttt{BETWEEN}\}$, while the `DAL` module predicts the ordering keyword $y \in \{\texttt{DESC}, \texttt{ASC}, \texttt{DESC LIMIT}, \texttt{ASC LIMIT}\}$.

3.2 Knowledge as Features

The adjective-noun phrasing knowledge is formulated as additional input features to existing models. Let X_K denote the knowledge feature, we rewrite the classifier for `OP` and `DAL` module as:

$$p(y) = \mathcal{F}(V \tanh(W[X : X_K])), \qquad (2)$$

where $[:]$ denotes the concatenation of feature vectors. The intuition is that even if X is unseen, the knowledge feature X_K will help to make the correct prediction. We will give an example to show how to construct the knowledge feature.

As the adjective-noun phrasing knowledge consists of two clusters, we use two fixed random vectors to represent the two clusters, denoted as polarity features. We define the knowledge feature as the attention weighted polarity features with knowledge masks.

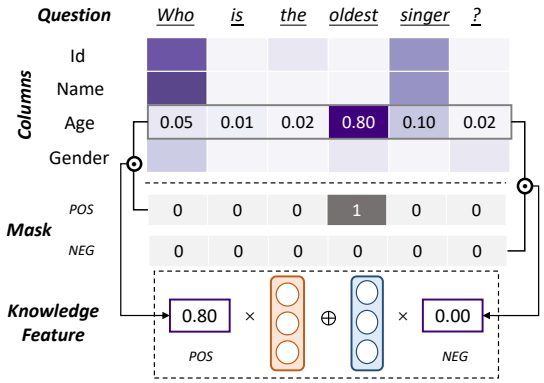

Figure 1: Knowledge as feature.

In Figure 1, the up palette shows the attention matrix between column names and question tokens. In syntaxSQLNet and SQLNet, the attention between column names and question tokens is defined by:

$$\alpha_{Q/CS} = \text{softmax}(H_Q W H_{CS}^{\top}), \qquad (3)$$

where H_{CS} and H_Q represent the hidden state of LSTM for column names and question tokens, respectively. Suppose that we are predicting the comparison operator for column `age` using the

DAL module. We first construct the positive and negative knowledge mask vectors, whose lengths are equal to the number of question tokens. In mask vectors, 1 means that the pair of the column name and the corresponding question token exists in phrasing knowledge. Then, we calculate the weight of positive and negative polarity features using the inner product ($\odot$) of knowledge mask vectors and the attention vector. For example, in Figure 1, the weight of positive and negative polarity features are 0.8 and 0, respectively. After that, we obtain the knowledge feature by the element-wise sum ($\oplus$) of the weighted positive and negative polarity features, denoted by $+_{\text{weighted}}$.

To obtain the mask vectors, we need to match the nouns in phrasing pairs with column names in the Spider dataset. For example, column names `birth date`, `birthdate`, `birth_date` and `date of birth` shoud be matched with the same noun *"date of birth"* in the phrasing knowledge. We heuristically define the matching score for column name n_1 and noun n_2 as:

$$\begin{aligned}\text{Score}(n_1, n_2) =& 0.2R_t(n_1, n_2) + 0.2R_c(n_1, n_2) \\ &+0.6\cos(\sum_{t_1 \in n_1} \text{idf}_{t_1} v_{t_1}, \sum_{t_2 \in n_2} \text{idf}_{t_2} v_{t_2}),\end{aligned} \quad (4)$$

where R_t and R_c are the token and char level fuzzy matching ratios, which is calculated by the output of fuzzywuzzy[8] divided by 100; and the last part is the cosine similarity between the inverse document frequency (idf) weighted sum of token embeddings, where the "document" refers to one column name in dataset. Given a column name in `OP` or `DAL` module, we select phrasing pairs whose noun matching score is ranked in top 10 as the knowledge to obtain the knowledge mask vectors.

Another straightforward way is to define the polarity feature as the knowledge feature directly, denoted as $+_{\text{direct}}$, where the polarity is determined by the attention weighted knowledge mask score. In the example above, the positive polarity feature will be considered as knowledge feature since the positive mask has a higher attention weighted score ($0.8 > 0$). We will evaluate both the $+_{\text{weighted}}$ and $+_{\text{direct}}$ knowledge features in the experiments.

4 Experiments

4.1 Data & Settings

For quality assurance, we manually choose 91 adjectives that have value polarities from top 200 frequent adjectives in Wikipedia. The adjectives with no value polarities like "important", "happy" and "reasonable" are filtered out. Then we use the adjective synonyms crawled from Bing Dictionary[9], the noun synonyms and adjective antonyms from WordNet (Miller, 1995) to cluster the adjective-noun pairs. We obtain 4,133 distinct phrasing pairs in 2,689 clusters. After that, we manually label the clusters with total sum of pair frequencies in top 50 or total number of pairs in top 50. Finally, we obtain 970 distinct adjective-noun pairs that covers about 79.65% usage among all pairs. We manually review all the 970 pairs and the accuracy is 87.8%. The pairs after manually revision are served as the phrasing knowledge.

For Spider dataset, we remove the nested queries and queries containing `JOIN`, because the performance of baseline models on complex queries is rather low, and we would more like to see the result that whether the comparison relation is correctly predicted given the right column. There are 4,490 questions after removing the complex queries, with 3,946 training and 544 development data. We manually review all the 4,490 questions and the corresponding tables, and find that there are 1,212 queries containing comparison relations with polarity, 668 of which require phrasing knowledge for model to understand. There are 728 distinct adjective-column pairs in all the 668 queries, which is determined as a ground-truth knowledge (denoted as *_gt*). After matching the mined adjective phrasing knowledge (970 adjective-noun pairs) with the Spider column names using Equation 4, we obtain 598 distinct pairs covering 82.14% (= 598/728) of the adjective-column pairs above.

To further evaluate effectiveness of phrasing knowledge, we re-split the training and development data of Spider, to ensure that the knowledge applied in the development set has not been seen or applied in the training set. The sizes of the training and development sets are 3,906 and 584, respectively. For model training, the dimension of knowledge feature is 50 with other parameters remaining the same as baseline models.

4.2 Phrasing Knowledge Examples

In the step of value polarity mining (described in Section 2.2), we obtain 758 word pairs with opposite polarities. For example, the set of opposite-

[8] https://github.com/seatgeek/fuzzywuzzy

[9] https://www.bing.com/dict

	Select	-AGG	Where	-OP	ρ_w	Group	-Having	ρ_g	Order	-DAL	ρ_o	Overall
Original Splitted												
SQLNet	62.7	64.9	40.7	45.0	90.4	58.1	73.1	79.5	62.4	66.9	93.3	39.9
+weighted	63.2	63.9	**40.8**	44.6	91.5	48.1	59.2	81.3	61.1	67.2	90.9	40.4
+direct	**65.0**	**65.9**	36.4	37.9	**96.0**	**60.0**	65.5	**91.6**	**64.4**	**68.9**	**93.5**	**42.8**
+weighted_gt	65.7	67.0	40.8	42.7	95.6	58.9	68.4	86.1	64.4	68.9	93.5	43.0
+direct_gt	65.8	67.3	40.3	44.8	90.0	54.2	69.3	78.2	65.9	70.5	93.5	42.1
syntaxSQL	62.1	64.0	30.1	38.8	77.6	66.9	69.4	96.4	46.3	54.8	84.5	39.5
+weighted	**65.0**	**67.6**	33.3	40.3	82.6	59.0	63.1	93.5	**49.1**	**58.7**	83.6	**41.0**
+direct	61.8	64.6	**39.5**	**46.7**	**84.6**	66.4	68.9	**96.4**	48.7	57.7	84.4	40.1
+weighted_gt	65.9	68.1	37.6	47.2	79.7	60.6	63.1	96.0	53.5	58.9	90.8	42.5
+direct_gt	63.5	64.8	30.4	39.0	77.9	64.4	69.5	92.7	50.6	52.9	95.7	41.2
Re-Splitted												
SQLNet	64.4	65.7	38.4	42.4	90.6	52.1	64.5	80.8	60.9	68.3	89.2	40.8
+weighted	63.2	63.9	**40.8**	**44.6**	91.5	48.1	59.2	81.3	61.1	67.2	90.9	40.4
+direct	**65.0**	**65.9**	36.4	37.9	**96.0**	**60.0**	**65.5**	**91.6**	**64.4**	**68.9**	**93.5**	**42.8**
+weighted_gt	65.2	66.7	37.4	42.3	88.4	54.2	69.2	78.3	63.2	73.1	86.5	39.0
+direct_gt	62.8	64.4	40.8	44.5	91.7	60.0	72.2	83.1	60.7	70.0	86.7	40.1
syntaxSQL	66.0	69.0	30.0	41.2	72.8	64.8	68.0	95.3	50.5	60.4	83.6	39.7
+weighted	61.6	63.5	36.6	41.1	89.1	56.9	62.2	91.5	60.2	**71.5**	84.2	38.2
+direct	65.7	67.1	**37.9**	**41.6**	**91.1**	59.5	**69.8**	85.2	**62.5**	71.3	**87.7**	**40.8**
+weighted_gt	67.5	69.2	33.2	45.0	73.8	67.7	69.3	97.7	48.0	58.6	81.9	41.6
+direct_gt	64.1	66.2	38.5	49.9	77.2	66.7	68.2	97.8	52.7	60.5	87.1	41.4

Table 2: F1-score on the original and re-split single tables in Spider dataset. ρ means the rate of certain F1-score wo/w comparison components, e.g. $\rho_w = \text{F1}_{\text{where}}/\text{F1}_{\text{-op}}$. We report the results of introducing ground-truth knowledge (*_gt*) which is an upper bound of our model. The **bold number** means the best result for the adjective phrasing knowledge, while the underline number means the best result for the ground truth knowledge.

polarity words for *age* is {*birth year, date of birth, birthday*}, and for *rank* is {*point, vote, score*}.

	Positive	Negative
age	old, eld, high	young, low
birth date	young	old, eld
date	fresh, new, recent, late	old, early
score	high	low
rank	low	high
price	costly, expensive high, pricy	inexpensive cheap, low

Table 3: Examples of value polarity.

Table 3 shows some examples of the adjective-noun phrasing knowledge. It can be seen that the value polarities are opposite for `age` and `birth date`, `score` and `rank`.

4.3 Results on Comparison Relation Prediction

Table 2 shows the F1-scores on the original and re-split Spider dataset. -OP, -Having, and -DAL represent `WHERE` component without `OP` module, `Group` component without `Having` module, and `ORDER BY` component without `DAL` module, respectively. The results show that with phrasing knowledge introduced, the performances are all significantly improved.

To further analyze the performance on comparison relation prediction, we compare `WHERE` and -OP, `Group` and -Having, and `Order` and -DAL, denoted as ρ_w, ρ_g, and ρ_o, respectively. Taking the results on `Where` component of syntaxSQLNet as an example, when the column is correctly predicted, the estimated precision of `OP` is about 72.8% ($\approx 30.0/41.2$). After introducing phrasing knowledge, it is improved to 91.1% ($\approx 37.9/41.6$). This is similar for `DAL` module (from 83.6% to 87.7%). It demonstrates that the adjective-noun phrasing knowledge can extend the capabilities of existing models on comparison relation prediction from training to unseen data.

5 Conclusion & Future Work

In this paper, we introduce the adjective-noun phrasing knowledge as feature to improve comparison relation prediction on unseen data. Experimental results show that our apporach achieves promising performance. For future work, we will further improve the quality of phrasing knowledge or incorporate other concept-level knowledge in text-to-SQL.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China (Grand Nos. U1636211, 61672081, 61370126).

References

Jianpeng Cheng, Zhongyuan Wang, Ji-Rong Wen, Jun Yan, and Zheng Chen. 2015. Contextual text understanding in distributional semantic space. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 133–142. ACM.

Gerard De Melo and Mohit Bansal. 2013. Good, great, excellent: Global inference of semantic intensities. *Transactions of the Association for Computational Linguistics*, 1:279–290.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 731–742.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 351–360.

Jonathan Herzig and Jonathan Berant. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1619–1629.

Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. 2016. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 75–76. International World Wide Web Conferences Steering Committee.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Josef Ruppenhofer, Michael Wiegand, and Jasper Brandes. 2014. Comparing methods for deriving intensity scores for adjectives. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 117–122.

Raksha Sharma, Arpan Somani, Lakshya Kumar, and Pushpak Bhattacharyya. 2017. Sentiment intensity ranking among adjectives using sentiment bearing word embeddings. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 547–552.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. 2012. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 481–492. ACM.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):63.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 588–594.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Yuchen Zhang, Panupong Pasupat, and Percy Liang. 2017. Macro grammars and holistic triggering for efficient semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.