Mention Extraction and Linking for SQL Query Generation

Jianqiang Ma*, **Zeyu Yan***, **Shuai Pang, Yang Zhang, Jianping Shen**
{majianqiang554,yanzeyu751,pangshuai550,zhangyang147,shenjianping324}@pingan.com.cn
AI Team, Ping An Life Insurance Company of China, Ltd

Abstract

On the WikiSQL benchmark, state-of-the-art text-to-SQL systems typically take a slot-filling approach by building several dedicated models for each type of slots. Such modularized systems are not only complex but also of limited capacity for capturing inter-dependencies among SQL clauses. To solve these problems, this paper proposes a novel extraction-linking approach, where a unified extractor recognizes all types of slot mentions appearing in the question sentence before a linker maps the recognized columns to the table schema to generate executable SQL queries. Trained with automatically generated annotations, the proposed method achieves the first place on the WikiSQL benchmark.

1 Introduction

Text-to-SQL systems generate SQL queries according to given natural language questions. Text-to-SQL technology is very useful as it can empower humans to naturally interact with relational databases, which serve as foundations for the digital world today. As a subarea of semantic parsing (Berant et al., 2013), text-to-SQL is known to be difficult due to the flexibility in natural language.

Recently, by the development of deep learning, significant advances have been made in text-to-SQL. On the WikiSQL (Zhong et al., 2018) benchmark for multi-domain, single table text-to-SQL, state-of-the-art systems (Hwang et al., 2019; He et al., 2019) can predict more than 80% of entire SQL queries correctly. Most of such systems take a sketch-based approach (Xu et al., 2018) that builds several specialized modules, each of which is dedicated to predicting a particular type of slots, such as the column in `SELECT`, or the filter value in `WHERE`. Such dedicated modules are complex and often fall short of capturing inter-dependencies among SQL sub-clauses, as each type of slots is modeled separately. To deal with these drawbacks, this paper formulates text-to-SQL as mention extraction and linking problems in a sequence labeling manner (Section 2). In this new formulation, the key to synthesizing SQL is to extract the *mentions* of SQL slots and the *relations* between them. Consider the question and its corresponding SQL query in example (1), with the headers in the schema being {LANE, NAME, NATIONALITY, SPLIT (50M), TIME}.

(1) a. **Question**: What is the total sum of 50m splits for Josefin Lillhage in lanes above 8?
b. **SQL**: `SELECT SUM` (Split (50m)) `FROM` some_table `WHERE` Name = 'Josefin Lillhage' `AND` Lane $>$ 8

We can see that many SQL elements, or *slots*, such as `column` names of SPLIT (50M) and LANE, `values` like "Josefin Lillhage" and 8, as well as operators $>$ are mentioned with words similar in form and/or meaning. Moreover, the relations between the slot mentions, such as <lanes, above, 8> forming a filter condition, are represented by proximity in linear order or other linguistic cues. Thus, the recognition of the mentions and their relations would mostly reconstruct the intended SQL query from natural language question.

To this end, we leverage one unified BERT-based (Devlin et al., 2019) extractor (Section 2.1) to recognize the slot mentions as well as their relations, from the natural language questions. The output of the extractor can be deterministically translated into pseudo SQLs, before a BERT-based linker (Section 2.2) maps the column mentions to the table headers to get executable SQL queries. A major challenge to the proposed method is the absence of manual annotation of mentions and relations. Thus

*Equal contributions.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 6936–6942,
November 16–20, 2020.

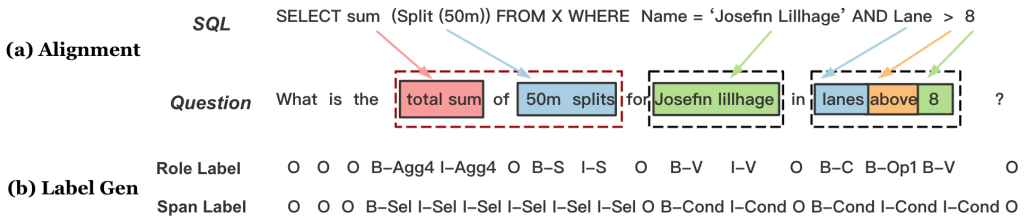

Figure 1: **Question sentence as mentions and relation of SQL slots.** (a) The mention of SQL slots (pink: the aggregation function, blue:columns, green: operators, yellow: values) and their relations (select relation and filter relation are shown in dashed rectangle). (b) Representing mentions with role labels and relations with span labels.

we propose an automatic annotation method (Section 2.4) based on aligning tokens in a SQL with corresponding question. Also, preliminary results show that the prediction of aggregation function (AGG) restricts model performance, which induces us to put forward AGG prediction enhancement (AE) method inspired by Brill (1995). Trained with such annotations and applied AE method, the proposed method can already achieves the first place on the WikiSQL benchmark.

The main contribution of this paper is the mention and relation extraction-based approach to text-to-SQL task. To the best of our knowledge, this is the *first* work that formulates the task as sequence labeling-based extraction plus linking, which enjoys the advantage of structural simplicity and interdependency awareness. In addition, we also propose an automatic method to generate annotations. Such annotations can be useful for developing novel methods for text-to-SQL, such as question decomposition-based approaches.

2 Method

2.1 Extractor

The extractor recognizes (1) *slot mentions*, including the SELECT column with aggregation function, WHERE columns with corresponding values and operators; and (2) *slot relations*, namely associating each WHERE column with its operator and value. Most of the SQL slots are mentioned in the question, as shown in Figure 1(a). As for the slot relations, note that the column, value and operator that form a filter condition relation usually appear in adjacency in the question, such as *in lanes above 8* in the example. Thus, the extraction of the relations is equivalent to the labeling of the corresponding *text span*. As shown in Figure 1(b), the extraction of mentions and relations can be represented by tagging each token in the question by a set of BIO labels. Formally, the label $l \in \{T \times \{B, I\}, O\}$, where $\times$ denotes the Cartesian product of T, the set of functional labels, and the set of positional label of {B, I}, where B and I means the beginning and the continuation of a particular annotation $t \in T$, respectively. The standing alone O label is assigned to tokens that are outside of any type of annotation of interest. For our task, we define two sets of labels: (a) the SQL *role labels* representing the slot mentions; (b) the *span labels* representing the slot relations, both of which are shown in Table 1. With these defined label set, the recognition of both slot mentions and slot relations are formulated as *sequence labeling*.

Role Type	F-Labels	Example
select column	S	*splits*: B-S
where column	C	*lanes*: I-C
value	V	*8*: B-V
agg. function	AGGi	*sum*: I-AGG4
operator	OPi	*above*:B-OP1
Span Type	**F-Label**	**Example**
SELECT span	Sel	*sum*:I-Sel
FILTER span	Cond	*lanes*:B-Cond

Table 1: **Labels for mention *roles* & relation *spans*.**

Extractor Model The model first encodes the question text and the table headers. As pre-trained language models such as BERT achieve state-of-the-art performance on various NLP tasks including sequence labeling, we adopt BERT to get contextualized representations for both role and span labeling. Similar to state-of-the-art methods for text-to-SQL such as SQLova (Hwang et al., 2019), we concatenate the question text along with the table header as input for BERT, in the form of $q_1, q_2, .., q_L$, [SEP], $c_{1,1}, c_{1,2}, ...,$ [SEP], $c_{2,1}, ..., c_{M,1}...$, where Q ($|Q| = L$) is the question while $C = c_1, .., c_M$ ($|C| = M$) are the table headers. Each header c_i may have multiple tokens, thus the 2-d indexes of $c_{i,j}$ being used.

Special SEP token is inserted between different headers c_i as well as between the question sentence Q and the first header c_1. As the labeling is w.r.t. the question sentence, the conditional random filed (CRF) (Lafferty et al., 2001) layer only is applied to the question segment. The full model is described as in equation (1), where BERT denotes the BERT model while CRF denotes a CRF layer.

$$\begin{aligned} Q^{\text{B}}; C^{\text{B}} &= \text{BERT}([Q; C]) \\ Q^{att} &= \text{Attention}(Q^{\text{B}}, C^{\text{B}}, C^{\text{B}}) + Q^{\text{B}} \\ L &= \text{CRF}(W^L Q^{att}) \end{aligned} \quad (1)$$

Before the BERT representations are fed to the CRF layer, they first go through an *attention layer* (Bahdanau et al., 2014), which encodes the question tokens with columns in the schema. The resulting representation is added to the original token representation in an element-wise manner. Finally, the resulting token representations are fed to the CRF layer, which yields the label sequence. As the two labeling tasks can benefit each other, we fine-tune BERT in a multi-task learning way.

2.2 Schema Linking as Matching

The column mentions in the question sentence often differ with the the canonical column names in the table schema in terms of string forms, as shown in Figure 1, where SPLIT (50M) is mentioned as *50m splits* and NAME is not mentioned at all. The latter case is *implicit mention* of column, as only the value for the column, *Josefin Lillhage*, appears in the question. Such case is challenging yet not uncommon. To convert mention and relation extraction results to SQL, we need a schema linking module to link explicit and implicit column mentions to its canonical column names in the table schema. Formally, we define the linker as a text matching model, i.e. estimating a function $f([C_i; span; Q]) \rightarrow \{0, 1\}$, where C_i is a header in the table schema, $span$ is the either an extracted column mention (for linking explicit column mention) or an extracted value v (for linking implicit column mention). Special tokens of [W] and [S] are used to distinguish SELECT spans from FILTER spans. Again, BERT is used as the underlying model for its state-of-the-art performance on text matching. The matching procedure can be described as in equation (2).

$$\begin{aligned} v_i^{\text{CLS}} &= \text{BERT}([span; C_i]) \\ P(i) &= \text{Sigmoid}(W v^{\text{CLS}}) \end{aligned} \quad (2)$$

2.3 AGG prediction enhancement

Analysis of preliminary results suggests that aggregation function (AGG) prediction is a bottleneck for our system, which is partly attributed to the findings by Hwang et al. (2019) that AGG annotations in WikiSQL have up to 10% of errors. In such case, as our extractor model has to take care of other types of slots, these extra constraints make it more challenging for our model to fit flawed data, compared with a dedicated AGG classifier, as in most SOTA methods. Another reason may be that no all the aggregation functions are grounded to particular tokens. Given the characteristic of the data and the possible limitation of the information extraction-based model, we improve the AGG results over the original model, using only simple association signals in the training data. To this end, we adopt transformation-based learning algorithm (Brill, 1995) to update the AGG predictions based on association rules in the form of "change AGG from x to x', given certain word tuple occurrences." Such rules are mined and ranked from the training data by the algorithm.

2.4 Automatic Annotation via Alignment

A challenge for training the extractor is that benchmark datasets have no role or span annotations. Since manual annotations are costly, we resort to automatic ways. The idea is to annotate mentions by aligning the SQL slots in the query to tokens in the question. Figure 1 depicts such alignments with arrows and colors. Specifically, the proposed method is a two-step procedure. The first step is *alignment*, which runs two pass of aligning. The first pass conducts exact and partial string match to recognize values and some of the columns, while the second pass aligns the remaining SQL slots, by training a statistical aligner with the training set of the data. For this purpose, we choose Berkeley aligner (Liang et al., 2006), which works by estimating the co-occurrence of tokens in the parallel corpora, which are the question-SQL pairs in our case. As statistical aligner can occasionally yield null-alignment for a few tokens, we use another unsupervised word and semantic similarity-based algorithm (Perez et al., 2020) to complement the missing alignments. The second step is *label generation*, where the roles are generated according to aligned elements, while the span labels are assigned by considering minimal text span that covers all the elements in a SELECT/WHERE clause.

3 Experiment

Dataset and Metric. We use the largest human-annotated text-to-SQL dataset, WikiSQL Zhong et al. (2018), which consists of 80,654 pairs of questions and human-verified SQL queries. Tables appeared either in train or dev set will never appear in the test set. Two metrics in Zhong et al. (2018) are adopt for evaluating the SQL query synthesis accuracy: (1) *Logical Form Accuracy*, denoted as LF, where LF = # of SQL queries with correct logic form / total # of SQL queries; and (2) *Execution Accuracy*, denoted as EX, where EX = # of SQL queries with correct execution / total # of SQL queries. Execution guidance decoding (EG) (Wang et al., 2018) is used, following previous work.

Implementation Details. We use StanfordNLP (Qi et al., 2018) for tokenization. The word embeddings are randomly initialized by BERT, and fine-tuned during the training. Adam is used (Kingma and Ba, 2014) to optimize the model with default hyper-parameters. We choose uncased BERT-base pre-trained model with default settings due to resource limitations. The training procedures follows Hwang et al. (2019). Codes are implemented in Pytorch 1.3 and will be made publicly available [1].

3.1 Results

We compare our method with notable models that have reported results on WikiSQL task, including Seq2SQL(Zhong et al., 2018), SQLNet(Xu et al., 2018), TypeSQL(Yu et al., 2018a), Coarse-to-Fine(Dong and Lapata, 2018), SQLova(Hwang et al., 2019), X-SQL(He et al., 2019) and HydraNet (Lyu et al., 2020) in Table 2. Without EG, our method with BERT-base outperforms most of existing methods, including SQLova with BERT-large and MT-DNN (Liu et al., 2019a)-based X-SQL, and ranks right after HydraNet, which is based on RoBerTa (Liu et al., 2019b) large. Lyu et al. (2020) shows that RoBERTa large outperform BERT large in their setting and Liu et al. (2019a) shows MT-DNN also outperforms BERT in many tasks. Despite disadvantage in underlying pre-trained language model, our model achieves competitive results.

For the results with the EG in Table 2, our method outperforms all the existing methods, including SQLova, X-SQL and HydraNet, leading to new state-of-the-art in the SQL accuracies in terms of both logic form and execution. Table 3 shows the slot type-wise results, where our method achieves new state-of-the-art results on the W_{col}, W_{val} and W_{op} accuracies. Since the operators and values are directly derived from the extractor, such results are evidence for the effectiveness of our extraction-based approach. Before applying AGG enhancement (*AE*), the bottleneck of our method is on AGG prediction. We close such gap with *AE* using only word co-occurrence features. The improved AGG accuracy also leads to the new state-of-the-art for the overall SQL results. A limitation of our sequence labeling-based approach is that it performs passably on some questions with nested span structures, as in the question "When does the train [arriving at [Bourne]$_{\text{DEST}}$ at 11.45]$_{\text{TIME}}$ departure?" Since sequence labeling captures flat structures in nature, such cases raise challenges similar to the situation in nested NER.

Model	Dev		Test	
	LF	EX	LF	EX
Seq2SQL	49.5	60.8	48.3	59.4
SQLNet	63.2	69.8	61.3	68.0
TypeSQL	68.0	74.5	66.7	73.5
Coarse-to-Fine	72.5	79.0	71.7	78.5
SQLova	81.6	87.2	80.7	86.2
X-SQL	83.8	89.5	83.3	88.7
HydraNet	83.6	**89.1**	83.8	**89.2**
this work - AE	81.1	86.5	81.1	86.5
this work	**84.6**	88.7	**84.6**	88.8
SQLova+EG	84.2	90.2	83.6	89.6
X-SQL+EG	86.2	92.3	86.0	91.8
HydraNet+EG	86.6	92.4	86.5	92.2
this work - AE$_{EG}$	85.8	91.6	85.6	91.2
this work$_{EG}$	**87.9**	**92.6**	**87.8**	**92.5**

Table 2: **Accuracy of previous and this work**.

Model	S_{col}	S_{agg}	$W_{no.}$	W_{col}	W_{op}	W_{val}
SQLova	96.8	90.6	98.5	94.3	97.3	95.4
X-SQL	97.2	91.1	**98.6**	95.4	97.6	96.6
HydraNet	**97.6**	91.4	98.4	95.3	97.4	96.1
ours-AE	**97.6**	90.7	98.3	**96.4**	**98.7**	**96.8**
ours	**97.6**	**94.7**	98.3	**96.4**	**98.7**	**96.8**
SQLova$_{EG}$	96.5	90.4	97.0	95.5	95.8	95.9
X-SQL$_{EG}$	97.2	91.1	**98.6**	97.2	97.5	97.9
HydraNet$_{EG}$	**97.6**	91.4	98.4	97.2	97.5	97.6
ours-AE$_{EG}$	**97.6**	90.7	98.3	**97.9**	**98.5**	**98.3**
ours$_{EG}$	**97.6**	**94.7**	98.3	**97.9**	**98.5**	**98.3**

Table 3: **Test accuracy for each slot type.**

[1] https://github.com/nl2sql/IE-SQL

Estimating Annotation Quality. The quality of automatic annotation can be estimated in an *oracle extractor* setting, where the automatically annotated labels, instead of the extractor prediction, are fed to the linker. In this setting, the logic form and execution accuracy on the dev set reaches 92.8% and 94.2%, respectively, which are the ceiling for our approach. Note that such ceiling is above the human-level accuracy reported in Hwang et al. (2019), suggesting that the quality of the automatic annotation is reasonably good.

4 Related Work

Semantic parsing (Berant et al., 2013) is to map natural language utterances to machine-interpretable representations, such as logic forms (Dong and Lapata, 2016), program codes (Yin and Neubig, 2017), and SQL queries (Zhong et al., 2018). Text-to-SQL is a sub-area of semantic parsing, which is widely studied in recent years. Earlier work (Dong and Lapata, 2016; Krishnamurthy et al., 2017; Zhong et al., 2018; Sun et al., 2018; Wang et al., 2018) follow a neural sequence-to-sequence paradigm (Sutskever et al., 2014) with attention mechanism (Bahdanau et al., 2014). Pointer networks (Vinyals et al., 2015) are also commonly adopted. These sequence-to-sequence approaches often suffer the "ordering issue" since they are designed to fit an ordered sequence, while the conditions in WHERE-clause are unordered in nature.

SQLNet (Xu et al., 2018) introduces sketch-based method, which decomposes the SQL synthesis into several independent classification sub-tasks, including select-aggregation/column and where-number/column/operator/value. Except where-value, which is usually predicted by a pointer network, all the other sub-tasks use their own dedicated classifiers to make predictions. These sketch-based models raise challenges in training, deployment and maintenance. Moreover, each sub-module solves its own classification problem, without considering the dependencies with SQL elements modeled by other sub-modules. Recent advances (Yu et al., 2018a; Dong and Lapata, 2018; Hwang et al., 2019; He et al., 2019) follow this approach and achieve comparative results on WikiSQL, mostly by using pre-trained language models as the encoder.

While our sequence labeling method is also based on pre-trained language model, it differs from state-of-the-art methods in that it explicitly extracts mentions from the questions and can benefit from inter-dependency modeling between extracted mentions. The mentions for values, operators and corresponding columns often appear in proximity in the question, thus the sequence labeling model can better capture their dependencies and benefits the recognition for all of them, as experiment results suggest. Furthermore, our extractor-linker architecture is also much simpler than sketch-based methods.

Recent trend (Krishnamurthy et al., 2017; Guo et al., 2019; Wang et al., 2020; Choi et al., 2020) in academia starts to shift to multi-table and complex queries setting of text-to-SQL, as in the Spider task (Yu et al., 2018b). State-of-the art methods on Spider typically fall into two categories: *grammar-based approach* (Guo et al., 2019; Wang et al., 2020), and *sketch-based approach*, such as RYANSQL (Choi et al., 2020) and RECPARSER (Zeng et al., 2020). The latter ones have slot prediction modules similar to SQLNet for the WikiSQL, while recursion modules are introduced to handle the generation of complex SQL sketches, a characteristic in Spider but absent in WikiSQL. At a high level, our method is along the same line of SQLNet-RYANSQL, yet differs with them, as our method extracts slots in a unified way rather than using dedicated modules to predict each slot type. We can extend our method to the Spider task by following existing sketch construction methods as in RYANSQL, while replacing their slot classification modules with our extractor-linker methods.

5 Conclusion and Future Work

Thanks to the simple, unified model for mention and relation extraction and its capacity for capturing inter mention dependencies, the proposed method proves to be a promising approach to text-to-SQL task. Equipped with automatic-generated labels and AGG enhancement method, our model achieves state-of-the-art results on the WikiSQL benchmark. Since the current automatic-generated annotations are still noisy, it is useful to further improve the automatic annotation procedure. We also plan to extend our approach to cope with multi-table text-to-SQL task Spider.

Acknowledgements

We thank Jun Xu, Muhua Zhu, Wanxiang Che and Longxu Dou as well as all the anonymous reviewers for their invaluable comments and suggestions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2020. RYANSQL: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *arXiv preprint arXiv:2004.03125*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. *ArXiv*, abs/1601.01280.

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)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-sql: reinforce schema representation with context. *ArXiv*, abs/1908.08113.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *ArXiv*, abs/1902.01069.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 104–111, New York City, USA. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke S. Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. 2020. Hybrid ranking network for text-to-sql. Technical Report MSR-TR-2020-7, Microsoft Dynamics 365 AI.

Ethan Perez, Patrick Lewis, Wen tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. *ArXiv*, abs/2002.09758.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal Dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Yibo Sun, Duyu Tang, Nan Duan, Jianshu Ji, Guihong Cao, Xiaocheng Feng, Bing Qin, Ting Liu, and Ming Zhou. 2018. Semantic parsing with syntax- and table-aware SQL generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 361–372, Melbourne, Australia. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks.

In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Chenglong Wang, Po-Sen Huang, Alex Polozov, Marc Brockschmidt, and Rishabh Singh. 2018. Execution-guided neural program decoding. *ArXiv*, abs/1807.03100.

Xiaojun Xu, Chang Liu, and Dawn Xiaodong Song. 2018. Sqlnet: Generating structured queries from natural language without reinforcement learning. *ArXiv*, abs/1711.04436.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

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)*, pages 588–594, New Orleans, Louisiana. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018b. 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, Brussels, Belgium. Association for Computational Linguistics.

Yu Zeng, Yan Gao, Jiaqi Guo, Bei Chen, Qian Liu, Jian-Guang Lou, Fei Teng, and Dongmei Zhang. 2020. RECPARSER: A recursive semantic parsing framework for text-to-sql task. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3644–3650. International Joint Conferences on Artificial Intelligence Organization. Main track.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *ArXiv*, abs/1709.00103.