Leveraging Explicit Lexico-logical Alignments in Text-to-SQL Parsing

Runxin Sun[1,2], **Shizhu He**[1,2], **Chong Zhu**[1,2], **Yaohan He**[3], **Jinlong Li**[3], **Jun Zhao**[1,2] and **Kang Liu**[1,2,4]

[1] National Laboratory of Pattern Recognition, Institute of Automation, CAS, Beijing, China
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3] AI Lab, China Merchant Bank, ShenZhen, 518057, China
[4] Beijing Academy of Artificial Intelligence, Beijing, 100084, China
sunrunxin2020@ia.ac.cn, {shizhu.he, chong.zhu}@nlpr.ia.ac.cn,
{heyh18, lucida}@cmbchina.com, {jzhao, kliu}@nlpr.ia.ac.cn

Abstract

Text-to-SQL aims to parse natural language questions into SQL queries, which is valuable in providing an easy interface to access large databases. Previous work has observed that leveraging lexico-logical alignments is very helpful to improve parsing performance. However, current attention-based approaches can only model such alignments at the token level and have unsatisfactory generalization capability. In this paper, we propose a new approach to leveraging explicit lexico-logical alignments. It first identifies possible phrase-level alignments and injects them as additional contexts to guide the parsing procedure. Experimental results on SQUALL show that our approach can make better use of such alignments and obtains an absolute improvement of 3.4% compared with the current state-of-the-art.

1 Introduction

Text-to-SQL parsing is the task of mapping natural language questions to executable SQL queries on relational databases (Zhong et al., 2017). It provides an easy way for common users unfamiliar with query languages to access large databases and has attracted great attention. Recently, *lexico-logical* alignments, which align question phrases to their corresponding SQL query fragments, have been proved to be very helpful in improving parsing performance (Shi et al., 2020). As shown in Figure 1, the token "competitor" should be aligned to "c1" in the SQL query. To capture such alignments, several attention-based models were proposed (Shi et al., 2020; Lei et al., 2020; Liu et al., 2021), which employ the attention weights among tokens to indicate the alignments. Specifically, they use an attention module to perform schema linking at the encoding stage (Lei et al., 2020; Liu et al., 2021), and may use another attention to align each output token to its corresponding input tokens at the decoding stage (Shi et al., 2020).

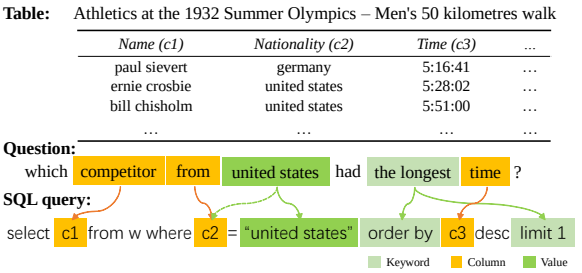

Figure 1: An example from SQUALL. Alignments belonging to the same type are marked with the same color.

However, we argue that the attention mechanism is not an appropriate way to capture and leverage lexico-logical alignments. It mainly has the following two problems. First, the standard attention can only model alignments at the token level rather than the phrase level, while there are many multi-granular, non-continuous alignments in the text-to-SQL task. For the example in Figure 1, "`order by ... limit 1`" is a SQL keyword pattern representing a superlative operation. However, the standard attention module can only align "`order`", "`by`", "`limit`", and "`1`" to "the longest" token by token, rather than regarding them as a whole. It may confuse the decoder and lead to the failure to generate this pattern correctly (Herzig and Berant, 2021). Second, traditional attention-based approaches are prone to overfitting the training data, which is harmful to the model's generalization capability. It is not only the *domain generalization* (Dong et al., 2019) but also the *compositional generalization* (Herzig and Berant, 2021). Specifically, the former refers to the generalization across different databases, while the latter refers to the ability to generate new structures composed of seen components.

To solve the aforementioned problems, we propose a neural parsing framework to leverage explicit lexico-logical alignments. Dong et al. (2019) have pointed out that if we align question tokens

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 2: Short Papers, pages 283 - 289

to columns or values in databases before parsing, it will help to improve the model's generalization among different domains (databases). Motivated by this, our framework consists of two steps. Specifically, we first implement a simple model to obtain possible lexico-logical alignments before parsing. While in the second step, we inject such alignments into a standard seq2seq parser by treating them as additional contexts, similar to "prompt information" or "evidence" in machine reading comprehension (Mihaylov and Frank, 2018; Tu et al., 2020; Niu et al., 2020). Moreover, to alleviate the negative effects on the parser caused by noise alignments, we propose a data augmentation method that adds noisy alignments during the training procedure. Experimental results on an open-released dataset, SQUALL (Shi et al., 2020), show that our framework achieves state-of-the-art performance and obtains an absolute improvement of 3.4% compared with existing attention-based models.

2 Preliminaries

2.1 Problem Definition

Here we consider the problem setting adopted by Shi et al. (2020). Formally, given a natural language question Q about a table T, our goal is to generate the corresponding SQL query Y, where the table consists of columns $\{c_1, \ldots, c_{|T|}\}$.

2.2 Base Parser

Our base parser is a standard seq2seq model. It generally follows the architecture proposed by Lin et al. (2020), which combines a BERT-based encoder with a sequential pointer-generator to perform an end-to-end parsing procedure.

Input Serialization and Encoder According to the definition above, an input X contains a length-n question $Q = q_1, \ldots, q_n$ and a table with m columns $T = \{c_1, \ldots, c_m\}$. We concatenate all the columns into a sequence for the table, where a unique token precedes each column to represent its type (e.g., text). Then we add two `[SEP]` tokens at both ends and append this sequence to the question. After adding a `[CLS]` token at the beginning, we get the input sequence in the following format:

$$X = \texttt{[CLS]}, Q, \texttt{[SEP]}, \texttt{[TYPE\#C1]}, c_1, \ldots, \texttt{[TYPE\#Cm]}, c_m, \texttt{[SEP]}$$

X is encoded with BERT (Devlin et al., 2019), followed by a bidirectional LSTM (bi-LSTM) to get the hidden representations $\boldsymbol{h}_X$. Then for the question part, we feed its representation to another bi-LSTM to obtain the encoding result $\boldsymbol{h}_Q$. Each column is represented by the vector of its corresponding type token.

Decoder Like Lin et al. (2020), We use an LSTM-based pointer-generator (See et al., 2017) enhanced with the attention mechanism as the decoder. Specifically, we use the final hidden state of the question encoder to initialize the decoder. At each step t, the decoder chooses one of the following three actions: generating a keyword from the vocabulary V, copying a token from the question Q, or copying a column from the table T.

3 Method

3.1 Framework Overview

As shown in Figure 2, our framework consists of two stages: *lexico-logical alignment prediction* (the upper left) and *alignment-enhanced parsing* (the bottom). At the first stage (*alignment prediction*), we identify possible lexico-logical alignments in the question before parsing. At the second stage (*alignment-enhanced parsing*), we inject these alignments into the parser so that it can make further completions and refinements based on them.

3.2 Lexico-logical Alignment Prediction

In this step, we implement a simple model to predict lexico-logical alignments of the input question. Specifically, we adopt a two-stage pipeline process: 1) identify question phrases that may have alignments; 2) predict their corresponding query fragments according to the types.

For the first stage, we classify the alignments into three types according to their corresponding query fragments: *keyword*, *column*, *value*. Specifically, ***keyword*** alignments map question phrases to query fragments composed of SQL keywords, while the other two types of alignments (***column*** and ***value***) map them to columns in databases. The only difference between ***column*** and ***value*** alignments is that the phrase part of a *value* alignment is also a value in the SQL query. Analogous to Named Entity Recognition (NER), we use sequence labeling to implement this process:

$$P(label_i \mid Q, T) = \text{softmax}(\text{MLP}([\boldsymbol{h}_i; \boldsymbol{c}_i])). \quad (1)$$

Here we apply the BIO labeling schema, classifying each token as one of the four types: *keyword*,

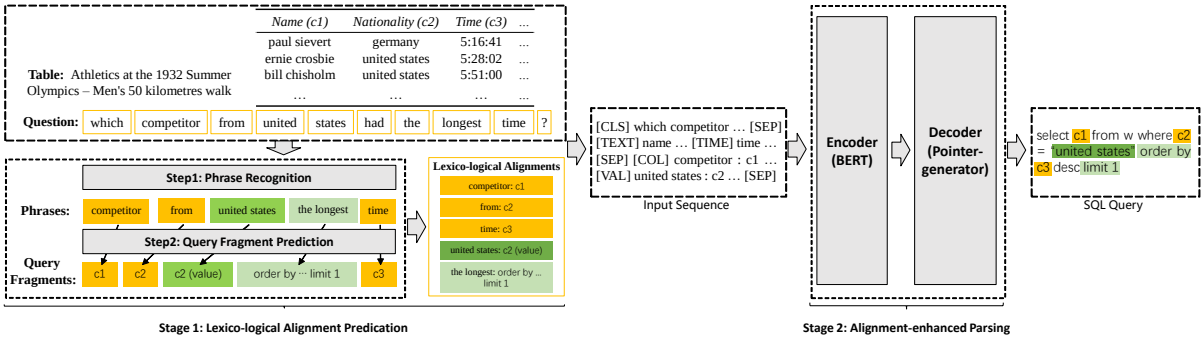

Figure 2: An illustration of our framework. It consists of two stages: (i) lexico-logical alignment prediction; (ii) alignment-enhanced parsing.

column, *value*, or *none*. We adopt the same structure as our base parser to encode the input sequence, and $\boldsymbol{h}_i$ is the hidden representation of the i-th token. Moreover, an attention module is used to get the column-aware question representation $\boldsymbol{c}_i$:

$$\boldsymbol{c}_i = \text{Attention}(\boldsymbol{h}_i, \boldsymbol{h}_C, \boldsymbol{h}_C), \tag{2}$$

where $\boldsymbol{h}_C$ are the representations of all the columns. Then we run a Multi-Layer Perceptron (MLP) by concatenating these two vectors as inputs to predict the i-th label.

For the second stage, we predict the query fragment corresponding to the phrase. Specifically, we can divide this process into the following two cases according to the type of phrase:

1) *Keyword*: We use a generation model to obtain keyword fragments corresponding to such phrases. In detail, we perform self-attention on the token representations of the phrase p to get the initial hidden state. Then we run an RNN model with attention to generate its corresponding keyword fragment:

$$y = \arg\max \prod_t P(y_t \mid y_{<t}, p). \tag{3}$$

2) *Column & Value*: In this case, we should link the phrase to its corresponding column. Intuitively, based on the attention matrix, we can directly get the column c^* that best matches the phrase p:

$$c^* = \arg\max_{c \in C} f(p, c) = \arg\max_{c \in C} \sum_{w \in p} f(w, c). \tag{4}$$

3.3 Alignment-enhanced Parsing

After getting all the lexico-logical alignments in a question, we then consider adding them to the parsing process. Naturally, we design their usages for both the encoding and decoding processes.

For the encoding stage, we treat alignments as additional contexts and add them to the input sequence. Concretely, we represent each alignment as a concatenation of the natural language phrase p and its corresponding query fragment f, where the two parts are separated by ":". Moreover, a unique token before each of them represents its type (*keyword*, *column* or *value*). Thus the format of the modified input sequence is as follows:

$$\begin{aligned} X^+ = &\texttt{[CLS]}, Q, \texttt{[SEP]}, \texttt{[TYPE\#C1]}, c_1, \ldots, \\ &\texttt{[TYPE\#Cm]}, c_m, \texttt{[SEP]}, \texttt{[TYPE\#A1]}, \\ &p_1, :, f_1, \ldots, \texttt{[TYPE\#An]}, p_n, :, f_n \texttt{[SEP]} \end{aligned}$$

In this way, the encoder based on a pre-trained language model can make good use of this information to help it better perform schema linking.

For the decoding stage, we also add alignments to the generation process. Specifically, we take its type token's hidden vector as the representation of each alignment, denoting it as $\boldsymbol{h}_A$. So at each step t, we compute the attention between the decoder hidden state $\boldsymbol{h}_t^D$ and the alignments:

$$\boldsymbol{a}_t = \text{Attention}(\boldsymbol{h}_t^D, \boldsymbol{h}_A, \boldsymbol{h}_A). \tag{5}$$

Afterward, we use the concatenation of $\boldsymbol{a}_t$ and the embedding of the previous token $\boldsymbol{e}_t$ as the decoder's input, injecting this information into the next step's hidden state:

$$\boldsymbol{h}_{t+1}^D = \text{LSTM}^D([\boldsymbol{e}_t; \boldsymbol{a}_t], \boldsymbol{h}_t^D). \tag{6}$$

3.4 Noisy Alignment Augmentation

As mentioned before, it is impossible to obtain a perfect model for alignment prediction. So if we use the annotated alignments to train the parser, and

Model	Dev		Test
	$\mathbf{ACC_{LF}}$	$\mathbf{ACC_{EXE}}$	$\mathbf{ACC_{EXE}}$
SEQ2SEQ$^+$ + BERT	44.7 ± 2.1	63.8 ± 1.1	51.8 ± 0.4
ALIGN + BERT	47.2 ± 1.2	66.5 ± 1.2	54.1 ± 0.2
LAP (ours)	47.0 ± 1.3	65.0 ± 1.2	53.0 ± 0.5
+ noisy alignment	**50.6 ± 1.0**	**68.3 ± 0.8**	**56.5 ± 0.3**

Table 1: Overall parsing results. LAP refers to our model. "+ noisy alignment" means our model training under the noisy alignment augmentation.

use the predicted alignments to make predictions, then there is an inconsistency between training and testing. It is precisely because of this inconsistency that the parser tends to trust the given alignments completely. In that case, wrong alignments may hurt the parsing performance.

To alleviate the negative effects on the parser caused by noise alignments, we propose a method based on data augmentation, that is, adding noisy alignments during the training procedure. Specifically, we use the model proposed in section 3.2 to predict alignments for the training examples through cross-validation. Obviously, these alignments are noisy. Then we integrate these predicted examples with the annotated examples and use them as the augmented training set of the parser.

4 Experiments

4.1 Dataset and Experimental Setup

We evaluate on SQUALL (Shi et al., 2020), a large-scale dataset based on WIKITABLEQUESTIONS (Pasupat and Liang, 2015). It contains 11,276 table-question-answer triplets, enriched with human-annotated logical forms and lexical-logical alignments.[1] We use the default dataset split provided by Shi et al. (2020), where they randomly shuffle the tables and divide them into five splits so that examples with the same table are in the same split.

For evaluation metrics, we employ the average logical form accuracy $\mathrm{ACC_{LF}}$ and execution accuracy $\mathrm{ACC_{EXE}}$,[2] following Shi et al. (2020). For model implementation, please refer to Appendix A for more details. It is worth noting that, unless otherwise stated, we only use the alignment annotations of the training set to train the alignment prediction model. While on the dev / test set, we use the predicted alignments as the parser's input.

[1]There are no such annotations for the test set.

[2]$\mathrm{ACC_{LF}}$ checks whether the logical form output exactly matches the target, while $\mathrm{ACC_{EXE}}$ compares the execution results.

Model	DB split	Query split	IID split
SEQ2SEQ + BERT	43.5	1.2	48.1
+ attention sup.	46.7 (+ 3.2)	1.6 (+ 0.4)	51.1 (**+ 3.0**)
LAP (w/o alignment)	46.6	2.7	52.1
+ noisy alignment	50.0 (**+ 3.4**)	3.5 (**+ 0.8**)	53.0 (+ 0.9)

Table 2: Parsing results ($\mathrm{ACC_{LF}}$) over different splits of SQUALL. "+ attention sup." refers to using alignment annotations to supervise the attention module. LAP (w/o alignment) refers to our model without alignments.[3]

Model	$\mathbf{ACC_{LF}}$ (Dev)	Δ
SEQ2SEQ + BERT	43.5	
+ oracle attention	66.3	+ 22.8
LAP (w/o alignment)	46.6	
+ keyword alignment	58.1	+ 11.5
+ column alignment	55.1	+ 8.5
+ value alignment	54.1	+ 7.5
+ oracle alignment (token)	71.9	+ 25.3
+ oracle alignment	**73.1**	**+ 26.5**

Table 3: Parsing results on the 0-split under the oracle setting. SEQ2SEQ + BERT refers to the base parser (Shi et al., 2020) with BERT embeddings.

4.2 End-to-end Parsing Performance

To evaluate the effectiveness of our model, we compare end-to-end parsing performance with existing attention-based models. The results are shown in Table 1. For the baselines, we select SEQ2SEQ$^+$ and ALIGN provided by Shi et al. (2020). The former uses the automatically derived exact-match features to supervise the attention modules, while the latter uses the alignment annotations instead.

From the results, we can observe that after combining the alignment prediction model proposed in section 3.2, our parser (LAP) achieves state-of-the-art performance on SQUALL. We believe the reason is that our approach identifies possible lexico-logical alignments before parsing so that the parser can leverage such explicit alignments and model them on the phrase level. Moreover, "LAP + noisy alignment" further outperforms "LAP". It illustrates that noise alignments do have negative effects on the parser, while our noisy alignment augmentation method can alleviate them effectively.

4.3 Our Model's Generalization Capability

To evaluate the advantages of our model's generalization capability, we further made different splits of SQUALL (Shi et al., 2020) and conducted experiments on them. Here we evaluate the model's generalization capability from two perspectives: *domain generalization* and *compositional generalization*. Specifically, *DB split* refers to the default

[3]Please refer to Appendix B for more details.

cross-DB setting of SQUALL, where databases appearing in the test set were not seen during training, and we use it to test the model's *domain generalization*. *Query split* is the setting proposed by Finegan-Dollak et al. (2018) to test the model's *compositional generalization*, where no query template (query after anonymization of database-related variables) appears in more than one set. As for the *IID split*, it means that the test case is not in the training set while its corresponding database is seen during training. We employ it as the control group.[4]

The experimental results are shown in Table 2. From the results, we can observe that our approach (+ noisy alignment) obtains more significant improvement on the DB split and the query split, while it is not as effective as the attention-based approach on the IID split. In particular, our approach achieves twice the improvement on the query split (0.8 vs. 0.4), even on a stronger base parser. These reveal that our approach is more effective when parsing across different databases (*domain generalization*) and different query templates (*compositional generalization*), which illustrates that our approach has better generalization capability.

4.4 The Effectiveness of our Parser on Leveraging Lexico-logical Alignments

To evaluate the effectiveness of our parser on the lexico-logical alignments utilization, we conducted experiments under the oracle setting, where we used alignment annotations instead of predictions for testing. Table 3 shows the results.[5]

From the results, we can observe that 1) our parser obtains more improvements when injecting alignments (+ oracle alignment) than the attention-based approach (+ oracle attention). It proves that our model could more effectively utilize the *lexico-logical* alignment information. 2) We also show the results when injecting different types of alignment in our model. The results show that keyword alignment, which is excluded from traditional schema linking, is a valuable type and is also helpful in improving parsing performance. 3) Modeling such alignments at the phrase-level is more effective than the token-level ("+ oracle alignment" vs. "+ oracle alignment (token)").

[4]Please refer to Appendix B for more details.

[5]Because Shi et al. (2020) did not provide the oracle results with BERT, we re-ran the open-source code (https://github.com/tzshi/squall) and got the results. Besides, due to the limitation of resources, we conducted them only on the 0-split of SQUALL instead of all five splits.

5 Conclusion

In this paper, we propose a neural parsing framework to leverage explicit lexico-logical alignments by treating them as additional contexts. Moreover, to alleviate the negative effects on the parser caused by noise alignments, we add noisy alignments during training inspired by data augmentation. Experimental results on SQUALL show that our framework achieves state-of-the-art performance compared with existing attention-based models.

Acknowledgments

This work is supported by the Natural Key R&D Program of China (No.2020AAA0106400), the National Natural Science Foundation of China (No.61922085, No.61976211) and the Key Research Program of the Chinese Academy of Sciences (Grant NO.ZDBS-SSW-JSC006). This research work was supported by the independent research project of National Laboratory of Pattern Recognition, the Youth Innovation Promotion Association CAS and Yunnan Provincial Major Science and Technology Special Plan Projects (No.202103AA080015).

References

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.

Zhen Dong, Shizhao Sun, Hongzhi Liu, Jian-Guang Lou, and Dongmei Zhang. 2019. Data-anonymous encoding for text-to-SQL generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5405–5414, Hong Kong, China. Association for Computational Linguistics.

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)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional general-

ization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, page 908–921, Online. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6943–6954, Online. Association for Computational Linguistics.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888, Online. Association for Computational Linguistics.

Qian Liu, Dejian Yang, Jiahui Zhang, Jiaqi Guo, Bin Zhou, and Jian-Guang Lou. 2021. Awakening latent grounding from pretrained language models for semantic parsing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1174–1189, Online. Association for Computational Linguistics.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832, Melbourne, Australia. Association for Computational Linguistics.

Yilin Niu, Fangkai Jiao, Mantong Zhou, Ting Yao, Jingfang Xu, and Minlie Huang. 2020. A self-training method for machine reading comprehension with soft evidence extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3916–3927, Online. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: an imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 8026–8037, Red Hook, NY, USA. Curran Associates Inc.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to SQL queries. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1849–1864, Online. Association for Computational Linguistics.

Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. 2020. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9073–9080.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

A Model Implementation Details

Our model is implemented in PyTorch (Paszke et al., 2019). For the BERT model, we fine-tune a `bert-base-uncased` model from the *Hugging Face's Transformers* library (Wolf et al., 2020). For the attention module, we use the standard dot-product attention function. We set all LSTMs to 1-layer and hidden size to 256. We use the Adam optimizer (Kingma and Ba, 2015) and clip gradients to 2.0. For the loss function, we choose cross-entropy for the classification task and label-smoothing for the generation task. We train our alignment prediction model for up to 10 epochs and SQL parser for 20 epochs. Both of them have an epoch for warm-up, and then the learning rate will decay linearly.

In terms of hyperparameter search, we turned the batch size (**8**, 16, 32), max learning rate (1e-3, **1e-4**), max BERT learning rate (5e-5, **2e-5**, 1e-5, 5e-6), and dropout (0.1, **0.2**, 0.3, 0.5). Due to the limitation of resources, we turned these parameters one by one instead of using grid search. The bolded values are a set of optimal parameters we found.

B Details of Different Splits of the SQUALL Dataset

We made three different splits of the SQUALL dataset: *IID split*, *DB split*, and *query split*, to explore the corresponding generalization capabilities of the model. It is worth noting that because this dataset is a single-table dataset (that is, each DB contains only one table), the cross-DB setting is essentially equal to the cross-table setting. The specific methods for obtaining these splits are as follows:

- ***IID split***: In order to ensure that tables in the test set are also in the training set, and the only difference between the two sets is that the included samples are different, we classify the samples according to their corresponding tables. For each category (i.e., table), we randomly select k (in this case, $k = 1$) samples, put them into the test set, and put the rest into the training set.

- ***DB split***: This is the default setting of the SQUALL dataset. Here we use the 0-split provided by Shi et al. (2020).

- ***query split***: Inspired by Finegan-Dollak et al. (2018), we substitute variables for table-related entities (i.e., columns and values) in each query in the dataset to obtain its corresponding query template, just like Shi et al. (2020) did. Similarly, we classify the samples according to their corresponding query templates. For each category (i.e., query template), all its samples can only be put into either the training set or the test set. It is worth noting that to examine the *compositional generalization* better, we sort the templates according to their frequency. Then we put the templates with higher frequency into the training set and the templates with lower frequency into the test set.

For the above three splits, we make the ratio of the training set and the test set approximately equal to 4:1, consistent with Shi et al. (2020).

C Details on Obtaining Token-level Alignments

To verify whether our approach can model alignments at the phrase level, we constructed token-level alignments to contrast with the original alignment annotations. Specifically, we imitated the attention mechanism and decomposed the alignments according to their types.

For *keyword* alignments, inspired by Shi et al. (2020), we align each keyword in the SQL query to all its corresponding tokens in the question. For the example in Figure 1, we align "`order`", "`by`", "`limit`", and "`1`" to "the longest" respectively. Then we obtain the following four alignments: "the longest: `order`", "the longest: `by`", "the longest: `limit`", and "the longest: `1`".

For the other two types of alignments: *column* and *value*, as mentioned in section 3.2, they both align question phrases to columns in databases. Analogous to schema linking through an attention module, we align each token in the question phrase to the corresponding column separately. For the example in Figure 1, we align "united" and "states" to column `c2` respectively instead of treating these two tokens as a whole.