Generating SQL Statements from Natural Language Queries: A Multitask Learning Approach

Chunqi Chen, Yunxiang Xiong, Beijun Shen*, Yuting Chen
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University, Shanghai, China
{15274521452, bruinx, bjshen, chenyt}@sjtu.edu.cn

***Abstract*—NL2SQL advocates an idea of helping engineers and/or end users generate SQL statements from natural language queries. However, it still remains a strong challenge in improving its precision and scalability. This paper introduces MultiSQL, a multitask deep learning approach to performing NL2SQL. MultiSQL unifies the task representations and trains a model in parallel on multiple tasks, including NL2SQL, machine translation, etc. It employs a multitask question-answering network for jointly learning all tasks and transferring knowledge among tasks. We have evaluated MultiSQL on two query datasets: WikiSQL (an open sourced dataset) and CnSQL (a Chinese dataset we created). The evaluation results clearly show the effectiveness of MultiSQL. In particular, the accuracies achieved by MultiSQL approximate those achieved by the state-of-the-art NL2SQL methods on WikiSQL, and its accuracy is 78%, which is 17% higher than the "Chinese2English + NL2SQL" method on CnSQL.**

***Index Terms*—NL2SQL, SQL statement generation, deep learning, multitask learning**

I. Introduction

With the rapid development of data engineering, industry engineers frequently perform data queries for data analyses and/or obtaining online reports. SQL is a popular and flexible language for querying data. In order to facilitate end users to perform data queries, some studies on NL2SQL, i.e., translating natural language queries into SQL statements, have been conducted.

So far two mainstreams of NL2SQL methods do exist. One mainstream is to use a *seq2seq* (sequence to sequence) model with neural networks for query generation [1], [2]. In particular, training a seq2seq model requires a standard format [2]. Meanwhile, a query result can have different logical forms, which reduces the effectiveness of the training process. *Seq2set* (sequence to set) is another mainstream, which divides a query into parts and generates them individually [3], [4]. A seq2set method is capable of processing the disorders of filtering conditions effectively, but it may miss internal dependencies in the input sequences.

The existing efforts have achieved remarkable results. However, they are still facing two main difficulties when applied in practice.

First, *many NL2SQL methods are only evaluated on the machine-generated datasets, making their effectiveness unclearly shown.* Indeed, most of the methods are evaluated on WikiSQL*–an open sourced dataset automatically extracted from Wikipedia. An NL2SQL method should be evaluated on both the machine- and the human-generated datasets [5].

Second, *many NL2SQL methods are not scalable, as they only focus on translating queries in English into SQL statements.* In case that queries in other languages are raised, they must at first be translated into descriptions in English, and then to SQL statements. The imprecision occurred during the translation stage can be propagated.

One solution to this is combining the language translation task with NL2SQL to seize the latent information in the queries and datasets. This paper presents MultiSQL, a multitask learning approach to NL2SQL. MultiSQL learns one model for multiple NLP tasks, including NL2SQL, machine comprehension, machine translation, etc. MultiSQL uses a TCR (Task-Content-Result) template to unify all tasks, and builds a multitask deep learning network for jointly learning all tasks and transferring knowledge among them.

This paper makes the next contributions:

1) **A deep learning approach**. We propose a general, scalable multitask deep learning approach, MultiSQL, to NL2SQL. MultiSQL trains a model for multiple NLP tasks, and takes three training strategies for accelerating sample efficient learnings and supporting knowledge transfers among tasks.
2) **A multitasking neural network**. We design a multitasking neural network with an encoder and a decoder. The encoder adopts dual coattention to represent the task and its content sequences, compressing all of this information with two BiLSTMs. The decoder with multi-pointer-generator utilizes attention over the content, task and previously output tokens to make decisions.
3) **Dataset and evaluation**. We have evaluated MultiSQL on WikiSQL and CnSQL. CnSQL is a Chinese SQL dataset we collected from real-world applications. Through transfer learning, the performance of NL2SQL is enhanced, achieving the logical form accuracy of 78.7% and the database execution accuracy of 86.1%. Furthermore, MultiSQL achieves logical form accuracy

*Corresponding author.
DOI reference number: 10.18293/SEKE2019-024

*https://github.com/salesforce/WikiSQL

of 78%, which is 17% higher than the "Chinese2English + NL2SQL" method.

II. Related Work

A. Code Generation and NL2SQL

Code generation is becoming a promising approach to improve the productivity of software development. At present, there are three mainstream technologies: generation from models using templates [6], program synthesis by logic specification, and code generation and recommendation via machine learning or information retrieval [7]. NL2SQL is one of its research hotspots in recent decades, and in particular, neural-network-based NL2SQL has achieved remarkable results.

Dong and Lapata [1] have introduced a seq2seq approach that uses the augmented pointer network to convert textual queries to logical forms. Zhong et al. [2] have published the WikiSQL dataset and proposed a seq2seq model with reinforcement learning. Xu et al. [3] have further improved the results by taking a seq2set model and an attentional model. Similarly, Guo and Gao [8] have developed tailored modules to process three components in SQL queries. A parallel work [4] obtained a high execution accuracy on WikiSQL for SQL statements belongs to the "select-aggregator-where" type. External knowledge bases are also employed for tagging question words. Sun et al. [9] have presented a generative learning model to replicate contents in column names, cells, or SQL keywords. They have also improved the generation of the WHERE clauses by leveraging the column-cell relations.

B. Multitask Learning

Multitask learning is a machine learning paradigm. It aims to improve the generalization performance of a task using many other related tasks. Multitask learning has been successfully applied in the domain of natural language processing.

Collobert et al. [10] have proposed a unified framework for processing multiple natural language tasks, including chunking and part-of-speech tagging. Hashimoto et al. [11] have presented a neural network for dependency analysis, semantic correlation, and natural language reasoning. Zero-shot translation can be achieved by multitasking in language translations [12], where seq2seq models use two or more encoders and decoders for translation, parsing, and image subtitles [13]. Through model modularization, the above approaches can be applied to image classification and speech recognition [14]. Learning this modularity can further alleviate task interruptions [15]. Through multitask learning, the model can be used to learn some generally purposed expressions, since simultaneous learning of relevant tasks can provide inductive bias. Performance can be improved due to knowledge transfers across tasks.

III. Approach

MultiSQL is a multitask-learning-based approach to NL2SQL. As Fig. 1 shows, it takes a deep learning model to learn multiple tasks simultaneously. The deep learning model consists of a dual coattention encoder and a multi-pointer-generator decoder.

A. A Multitask QA Network

In MultiSQL, every task is formulated as a QA (Question Answering) task. Multiple tasks are trained jointly using a deep learning model, and knowledge is shared among tasks.

Task Unification. We design a TCR (Task-Content-Result) template to unify all the tasks. Each task instance is described with a *task*, *content*, and *result*, as Fig. 2 shows. During training, MultiSQL takes three sequences as its inputs: a content C with l tokens, a task Q with m tokens, and a result A with n tokens. Each token is represented by a d_{emb}-dimensional embedding.

Dual Coattention and Multi-Pointer. A task formed by the TCR template often contains key information that constrains the search space. MultiSQL uses dual coattention [16] for presenting conditions for both sequences, compresses the information with two BiLSTMs, applies self-attention [17] to collect long-distance dependencies, and then uses two BiLSTMs to get representations of the task and its content. The multi-pointer-generator [18] decoder uses attentions over the content, task, and previously output tokens to make a decision: copying from the content, copying from the task, or generating from a limited vocabulary.

Cross-task Transfer. MultiSQL facilitates sample-efficient learning and knowledge transfers among tasks. Three multitask collaborative training strategies can be used: *joint learning* (training all tasks jointly), *curriculum learning* (training simple tasks first), and *anti-curriculum learning* (training hard tasks first).

B. Encoder

MultiSQL adopts a deep stack-based recurrent neural network with a collaborative- and self-attention mechanism to generate the content embedding and the task embedding. The encoder takes six steps in the encoding process:

Step 1. Independent Encoding. A linear layer projects the input matrices onto a common d-dimensional space:

$$CW_1 = C_{proj} \in \mathbb{R}^{l \times d} \qquad QW_1 = Q_{proj} \in \mathbb{R}^{m \times d}$$

The projected representations are fed into a shared BiLSTM $BILSTM_{ind}$ to get the independent encoded representations $C_{ind} \in \mathbb{R}^{l \times d}$ and $Q_{ind} \in \mathbb{R}^{l \times d}$.

Step 2. Alignment. The encoder obtains the coattended representations by aligning encoded representations of each sequence. The alignments are obtained by normalizing dot-product similarity scores between the content sequence and the task sequence:

$$softmax(C_{ind}Q_{ind}^{\top}) = S_{cq} \qquad softmax(Q_{ind}C_{ind}^{\top}) = S_{qc}$$

Step 3. Dual Coattention. These alignments are used to compute weighted summations of a token in one sequence and a relevant token in another sequence:

$$S_{cq}^{\top}C_{ind} = C_{sum} \qquad S_{qc}^{\top}Q_{ind} = Q_{sum}$$

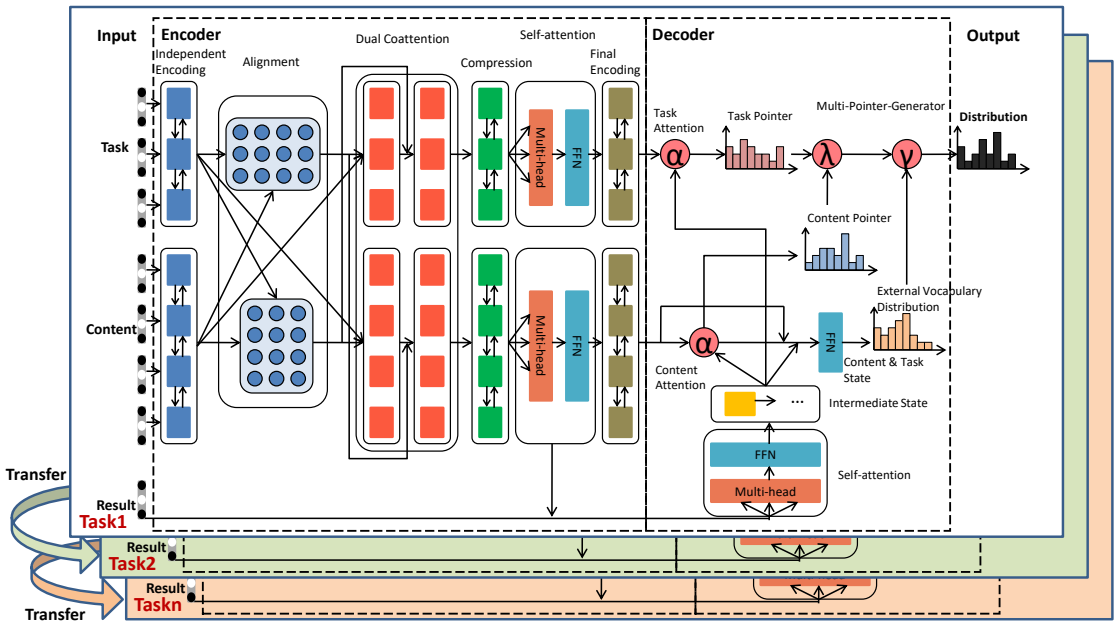

Fig. 1: An overview of the MultiSQL approach.

Task	Content	Result
Translate Chinese to English	告诉我南澳有哪些注意要点	Tell me what the notes are for South Australia
Generate SQL statements from neural text in English	The table has column names... Tell me what the notes are for South Australia	SELECT notes from table WHERE 'Current Slogan' = 'South Australia

Fig. 2: Some TCR examples.

The coattended representations use the same weights to transfer information gained from alignments back to the original sequences:

$$S_{qc}^{\top} C_{sum} = C_{coa} \qquad S_{cq}^{\top} Q_{sum} = Q_{coa}$$

Step 4. Compression. In order to compress embeddings from dual coattention back to the more manageable dimension d, we concatenate all of the four representations and feed them into separate BiLSTMs:

$$BiLSTM_{comC}([C_{proj}; C_{ind}; Q_{sum}; C_{coa}]) = C_{com}$$
$$BiLSTM_{comQ}([Q_{proj}; Q_{ind}; C_{sum}; Q_{coa}]) = Q_{com}$$

Step 5. Self-attention. We use multi-head, scaled dot-product attention [17] to capture long distance dependencies within each sequence.

$$Attention(\widetilde{X}, \widetilde{Y}, \widetilde{Z}) = softmax\{\frac{\widetilde{X}\widetilde{Y}^{\top}}{\sqrt{d}}\}\widetilde{Z}$$
$$MultiHead(\widetilde{X}, \widetilde{Y}, \widetilde{Z}) = [h_1; \cdots; h_p]W^o,$$
$$\text{where } h_j = Attention(\widetilde{X}W_j^{\widetilde{X}}, \widetilde{Y}W_j^{\widetilde{Y}}, \widetilde{Z}W_j^{\widetilde{Z}})$$

All transformations in the above equations are linear such that the multi-head attention representations maintain dimensionality as d.

$$MultiHead_C(C_{com}, C_{com}, C_{com}) = C_{mha}$$
$$MultiHead_Q(Q_{com}, Q_{com}, Q_{com}) = Q_{mha}$$

We then use the projected, residual feedforward networks (FFNs). The FNNs are equipped with ReLU activations and layer normalization on the inputs and outputs (with parameters $U \in \mathbb{R}^{d \times f}$, $V \in \mathbb{R}^{f \times d}$):

$$FFN(X) = \max(0, XU)V + X$$
$$FFN_C(C_{com} + C_{mha}) = C_{self} \in \mathbb{R}^{l \times d}$$
$$FFN_Q(Q_{com} + Q_{mha}) = Q_{self} \in \mathbb{R}^{m \times d}$$

Step 6. Encoding. Finally, we feed C_{self} and Q_{self} into two BiLSTMs $BiLSTM_{finC}$ and $BiLSTM_{finQ}$ to get the representations $C_{fin} \in \mathbb{R}^{l \times d}$ and $Q_{fin} \in \mathbb{R}^{m \times d}$, respectively.

C. Decoder

MultiSQL's decoder takes four steps in decoding:

Step 1. Self-attention. The decoder starts by projecting the answer embeddings onto a d-dimensional space:

$$AW_2 = A_{proj} \in \mathbb{R}^{n \times d}$$

Since recurrence and convolution are not contained in this step, we add positional encodings to A_{proj}:

$$A_{proj} + PE = A_{ppr} \in \mathbb{R}^{n \times d},$$
$$\text{where } PE[t,k] = \begin{cases} sin(\frac{t}{10000^{\frac{k}{2d}}}) & k \text{ is even;} \\ cos(\frac{t}{10000^{\frac{k-1}{2d}}}) & \text{Otherwise.} \end{cases}$$

We use self-attention so that the decoder is aware of previous outputs and the content to prepare for the next output. A residual FFN layer is applied to the content:

$$MultiHead_A(A_{ppr}, A_{ppr}, A_{ppr}) = A_{mha} \in \mathbb{R}^{n\times d}$$
$$MultiHead_AC((A_{mha} + A_{ppr}), C_{fin}, C_{fin}) = A_{ac} \in \mathbb{R}^{n\times d}$$
$$FFN_A(A_{ac} + A_{mha} + A_{ppr}) = A_{self} \in \mathbb{R}^{n\times d}$$

Step 2. Getting Intermediate Decoder State. We next apply a standard LSTM with attention to get a recurrent content state $\widetilde{c}_t$ for time-step t. The LSTM produces an intermediate state h_t using the previous answer word A^{t-1}_{self} and recurrent content state:

$$LSTM([A^{t-1}_{self}; \widetilde{c}_{t-1}], h_{t-1}) = h_t \in \mathbb{R}^d$$

Step 3. Task and Content Attention. This intermediate state is used to get attention weights α^C_t and α^Q_t, allowing the decoder to focus on encoded information of the time step t:

$$softmax(C_{fin}(W_2 h_t)) = \alpha^C_t \in \mathbb{R}^l$$
$$softmax(Q_{fin}(W_3 h_t)) = \alpha^Q_t \in \mathbb{R}^m$$

Content representations are combined with these weights and fed through an FFN with $tanh$ activation to form the content state and the task state:

$$tanh(W_4[C^\top_{fin}\alpha^C_t; h_t] = \widetilde{c}_t \in \mathbb{R}^d$$
$$tanh(W_5[Q^\top_{fin}\alpha^Q_t; h_t] = \widetilde{q}_t \in \mathbb{R}^d$$

Step 4. Multi-Pointer-Generator. The model may generate tokens that are not in the task or the content. Thus additional vocabulary tokens v are accessed. We obtain distributions over tokens in the task, content, and the external vocabulary:

$$\sum_{i:c_i=w_t} (\alpha^C_t)_i = p_c(w_t) \in \mathbb{R}^n$$
$$\sum_{i:q_i=w_t} (\alpha^Q_t)_i = p_q(w_t) \in \mathbb{R}^m$$
$$softmax(W_v\widetilde{c}_t) = p_v(w_t) \in \mathbb{R}^v$$

Two scalar tune the importance of each distribution in determining the output distribution:

$$\sigma(W_{pv}[\widetilde{c}_t; h_t; (A_{self})_{t-1}]) = \gamma \in [0, 1]$$
$$\sigma(W_{cq}[\widetilde{q}_t; h_t; (A_{self})_{t-1}]) = \lambda \in [0, 1]$$
$$\gamma p_v(w_t) + (1-\gamma)[\lambda p_c(w_t) + (1-\lambda)p_q(w_t)] = p(w_t)$$

A token-level negative log-likelihood loss is used throughout the training process: $\mathcal{L} = -\sum^T_t \log p(a_t)$.

IV. Experiments

We have implemented MultiSQL and evaluated it on several datasets. The evaluation is designed to answer the following research questions:

- **RQ1.** Is MultiSQL more effective than singletask learning methods?
- **RQ2.** Can MultiSQL be used to generate SQL statements from queries in other languages (Chinese in this study)?

A. Setup

The experiment contains ten NLP tasks, each of which is evaluated using one dataset. The ten datasets used in the evaluation are listed in Table I. In particular, NL2SQL was evaluated on WikiSQL (an open sourced dataset) and CnSQL (a Chinese dataset we created). CnSQL[†] includes 1534 pairs of Chinese queries and SQL statements. It was collected by the human engineers from end-user queries in a Chinese HR outsourcing platform (ezhiyang.com).

Several commonly used metrics are chosen: logical form accuracy [2] for NL2SQL, BLEU [19] for Chinese-English translation, and F1-Score for machine comprehension.

B. Results for RQ1

The pointer-generator seq2seq (S2S) model [18] is selected for comparison. The results for comparing the singletask and the multitask learnings are shown in Table I. Here, (w/SAtt), (+CAtt), (+QPtr), and (+ACurr) represent the S2S model with an adjunction of the self-attention mechanism, the dual co-attention mechanism, the task pointer, and the anti-curriculum learning strategy, respectively.

1) Singletask Learning

The self-attentive (w/SAtt) mechanism [17] increases the capacity of the S2S model of integrating information from the task and the content. It improves the performance on most of the tasks. For WikiSQL, this model approximates the state-of-the-art (72.4%), but it does not use a structured approach.

We supplement the model with a coattention mechanism (+CAtt) next. The performances on part of the tasks are improved while decrease on the others and significantly reduce on MNLI and MWSC. For these two tasks, since the S2S baselines have the content concatenated to the task, the pointer generator mechanism can copy words directly from the input. In case that the task and the content are separated, the effectiveness of the model may be reduced.

Thereafter, we add a task pointer (+QPtr) to the network, which boosts the performance on MNLI and MWS. It also improves performance on SQuAD to 75.5% in nF1, which matches the performance of the first wave of SQuAD models that utilize direct span supervision [16].

When tested on WikiSQL, the final model achieves logical form accuracy of 72.6% and database execution accuracy of 80.4%.

2) Multitask Learning

During multitask learning, the S2S model performs worse on many tasks than the singletask model, with a total score of 483.6 points. However, the performance of the model is getting better and better after the combination of (w/SAtt), (+CAtt), and (+QPtr) sequentially. The score finally reaches to 584.7 points.

Performances on tasks requiring heavy use of the external vocabulary drop more than 50% from the S2S baselines until (+QPtr) is added. In addition to a coattended content, the task pointer utilizes a coattended task, allowing task information

[†]https://github.com/SimonCqChen/CnSQL

TABLE I: Results for the singletask and multitask learnings.

Task	Dataset	Singletask				Multitask				
		S2S	w/SAtt	+CAtt	+QPtr	S2S	w/SAtt	+CAtt	+QPtr	+ACurr
NL2SQL	WikiSQL	60.0	**72.4**	72.3	72.6	45.8	64.8	72.9	74.0	**78.7**
Machine Translation	IWSLT	25.0	23.3	26.0	25.5	14.2	23.6	29.0	26.1	29.7
Machine Comprehension	SQuAD	48.2	68.2	74.6	**75.5**	47.5	66.8	71.8	70.8	74.3
Text Summarization	CNN/DM	19.0	20.0	25.1	24.0	25.7	14.0	15.7	23.9	24.6
Natural Language Inference	MNLI	67.5	68.5	34.7	72.8	60.9	69.0	70.4	70.5	69.2
Sentiment Classification	SST	86.4	86.8	86.2	88.1	85.9	84.7	86.5	86.2	86.4
Semantic Role Labeling	QA-SRL	63.5	67.8	74.8	75.2	68.7	75.1	76.1	75.8	77.6
Relationship Extraction	QA-ZRE	20.0	19.9	16.6	15.6	28.5	31.7	28.5	28.0	34.7
Goal Oriented Dialogue	WOZ	85.3	86.0	86.5	84.4	84.0	82.8	75.1	80.6	84.1
Semantic Parsing	MWSC	43.9	46.3	40.4	52.4	52.4	43.9	37.8	48.8	48.4
Total Score		-	-	-	-	483.6	566.4	543.8	584.7	**607.7**

to flow directly into the decoder. The main reason is that it is easy for the model to decide, if it can directly access the task, whether generating output tokens is more appropriate than copying.

By multitask joint learning with anti-curriculum strategy, the logical form accuracy of NL2SQL reaches 78.7%, and the database execution accuracy gets 86.1%.

C. Results for RQ2

The baseline for comparison is NL2SQL with Google's translator. The results show that MultiSQL obtains logical form accuracy of 78% on CnSQL. It is 17% higher than that of the baseline. The results also indicate that MultiSQL performs well both on the human- and the machine-generated datasets.

V. Conclusion

MultiSQL is a multitask learning approach to generating SQL statements from natural language queries. MultiSQL leverages a deep learning model for jointly learning multiple tasks such that the performance of NL2SQL can get improved. The evaluation results show the effectiveness of MultiSQL. It achieves accuracies that approximate those of the state-of-the-art methods and outperforms the existing methods for translating Chinese queries into SQL statements.

In the future, we will improve MultiSQL by introducing other state-of-the-art language models. We also plan to feed domain knowledge into MultiSQL, expecting that the deep learning model can be further enhanced.

VI. Acknowledgement

This research was sponsored by the National Key Research and Development Program of China (Project No. 2018YFB1003903), National Nature Science Foundation of China (Grant No. 61472242 and 61572312), and Shanghai Municipal Commission of Economy and Informatization (No. 201701052).

References

[1] L. Dong and M. Lapata, "Language to logical form with neural attention," *arXiv preprint arXiv:1601.01280*, 2016.

[2] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[3] X. Xu, C. Liu, and D. Song, "Sqlnet: Generating structured queries from natural language without reinforcement learning," *arXiv preprint arXiv:1711.04436*, 2017.

[4] T. Yu, Z. Li, Z. Zhang, R. Zhang, and D. Radev, "Typesql: Knowledge-based type-aware neural text-to-sql generation," *arXiv preprint arXiv:1804.09769*, 2018.

[5] C. Finegan-Dollak, J. K. Kummerfeld, L. Zhang, K. Ramanathan, S. Sadasivam, R. Zhang, and D. Radev, "Improving text-to-sql evaluation methodology," *arXiv preprint arXiv:1806.09029*, 2018.

[6] F. Mao, X. Cai, B. Shen, Y. Xia, and B. Jin, "Operational pattern based code generation for management information system: An industrial case study," in *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 425–430, IEEE, 2016.

[7] S. Zhou, H. Zhong, and B. Shen, "Slampa: Recommending code snippets with statistical language model," in *The 25th Asia-Pacific Software Engineering Conference (APSEC)*, 2018.

[8] T. Guo and H. Gao, "Bidirectional attention for sql generation," *arXiv preprint arXiv:1801.00076*, 2017.

[9] Y. Sun, D. Tang, N. Duan, J. Ji, G. Cao, X. Feng, B. Qin, T. Liu, and M. Zhou, "Semantic parsing with syntax-and table-aware sql generation," *arXiv preprint arXiv:1804.08338*, 2018.

[10] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. Aug, pp. 2493–2537, 2011.

[11] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher, "A joint many-task model: Growing a neural network for multiple nlp tasks," *arXiv preprint arXiv:1611.01587*, 2016.

[12] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, *et al.*, "Google's multilingual neural machine translation system: Enabling zero-shot translation," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 339–351, 2017.

[13] M.-T. Luong, Q. V. Le, I. Sutskever, O. Vinyals, and L. Kaiser, "Multi-task sequence to sequence learning," *arXiv preprint arXiv:1511.06114*, 2015.

[14] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit, "One model to learn them all," *arXiv preprint arXiv:1706.05137*, 2017.

[15] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard, "Learning what to share between loosely related tasks," *arXiv preprint arXiv:1705.08142*, 2017.

[16] C. Xiong, V. Zhong, and R. Socher, "Dynamic coattention networks for question answering," *arXiv preprint arXiv:1611.01604*, 2016.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[18] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," *arXiv preprint arXiv:1704.04368*, 2017.

[19] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318, Association for Computational Linguistics, 2002.