AdaNSP: Uncertainty-driven Adaptive Decoding in Neural Semantic Parsing

Xiang Zhang[1,2], **Shizhu He**[2], **Kang Liu**[1,2], **Jun Zhao**[1,2]
[1]University of Chinese Academy of Sciences, Beijing, 100049, China
[2]National Laboratory of Pattern Recognition (NLPR), Institute of Automation
Chinese Academy of Sciences, Beijing, 100190, China
{xiang.zhang, shizhu.he, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Neural semantic parsers utilize the encoder-decoder framework to learn an end-to-end model for semantic parsing that transduces a natural language sentence to the formal semantic representation. To keep the model aware of the underlying grammar in target sequences, many constrained decoders were devised in a multi-stage paradigm, which decode to the sketches or abstract syntax trees first, and then decode to target semantic tokens. We instead to propose an adaptive decoding method to avoid such intermediate representations. The decoder is guided by model uncertainty and automatically uses deeper computations when necessary. Thus it can predict tokens adaptively. Our model outperforms the state-of-the-art neural models and does not need any expertise like predefined grammar or sketches in the meantime.

1 Introduction

Semantic Parsing (SP) maps a natural language utterance into a formal language, which is crucial in abundant tasks, such as question answering (Zettlemoyer and Collins, 2005, 2007) and code generation (Yin and Neubig, 2017). The prevailing neural semantic parsers view semantic parsing as a sequence transduction task, and adopt the encoder-decoder framework similar to machine translation.

The distinguishing difference of semantic parsing, however, is in its target sequences, which are token sequences of well-formed semantic representations. SQL language and lambda expressions are typical examples of SP targets. The "`SELECT..FROM..WHERE`" pattern in SQL and the paired parentheses in lambda expressions are consequences of underlying grammars. However, standard Seq2Seq models ignore the patterns and may give ill-formed results.

To better model the grammatical and semantical constraints, many decoding methods were devised. Dong and Lapata (2018) proposed to generate tokens of an intermediate sketch first, followed by decoding into final formal targets. Others chose to gradually build abstract syntax trees using a transition-based paradigm, and tokens are generated at the tree leaves or in the middle of the transitions (Krishnamurthy et al., 2017; Chen et al., 2018; Yin and Neubig, 2018). There are also some decoders comprised of several submodules which are intended to generate different parts of the semantic output, respectively (Yu et al., 2018a,b). However, the aforementioned methods still have the following key issue. They explicitly require the expertise to design intermediate representations or model structures, which is not ideal or acceptable for scenarios with Domain Specific Languages (DSL) or new representations because of domain alterations and the incompleteness of the expert knowledge.

To follow the successful idea and overcome the above issue, we introduce a novel adaptive decoding mechanism. Inspired by adaptive computing (Graves, 2016), pervasive tokens in training data will be generated immediately with no doubt. But for tokens seen less often, the model may be pondering and less confident, and it will be better to carry out more computations. In this way, it is unnecessary to pre-build any intermediate supervision for training, such as preprocessed sketches (Dong and Lapata, 2018) and predesigned grammars (Yin and Neubig, 2018), which must be manually redesigned for an unseen kind of target language. Furthermore, we use the model uncertainty estimates to reflect its prediction confidence. Although different uncertainty estimates have been explored in semantic parsing (Dong et al., 2018), we use Dropout (Srivastava et al., 2014) as the uncertainty signal (Gal and Ghahramani, 2016) due

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4265–4270
Florence, Italy, July 28 - August 2, 2019.

to its simplicity, and use policy gradient algorithm to guide the model search.

Our contributions are thus three-fold.

- We introduce the adaptive decoding mechanism into semantic parsing, which is well rid of intermediate representations and easily adaptable to new target languages.
- We adopt uncertainty estimates to bias the decoder search, which is not covered in architecture searching literature to our best knowledge.
- Our model outperforms the state-of-the-art neural models without other intermediate supervisions.

2 Uncertainty-driven Adaptive Decoding Model

Our semantic parser is learned from pairs of natural language sentences and formal semantic representations. Let $x = \{x_1, x_2, \ldots, x_m\}$ denote the words in an input sentence, and $y = \{y_1, y_2, \ldots, y_n\}$ be the tokens of the corresponding target lambda expression.

2.1 Adaptive Decoding Model

We first introduce the general model for adaptive decoding. In general, the model consists of an encoder, a decoder, a halting module, and the attention mechanism.

Encoder. Input words x are first embedded using an embedding matrix $\mathbf{W}_x \in \mathbb{R}^{d \times |V_x|}$, where d is the dimension of embedded vectors and V_x is the set of all input words. We use a stacked two-layer BiLSTM to encode the input embedding. Hidden states from both direction at the same position of the second layer are concatenated as final encoder outputs $\{h_1, \ldots, h_m\}$.

Decoder. We stack two LSTM cells as one basic decoding unit. Similarlly, we use a matrix to embed target tokens y, $\mathbf{y}_i = \mathbf{W}_y \mathbf{o}(y_i)$. The token embedding will serve the input of the decoding cell.

$$s_t = f_{\texttt{LSTM}}\left(\left[\mathbf{y}_t; c_t^e; c_t^d; \texttt{flag}\right], s_{t-1}\right) \quad (1)$$

where $[\cdot;\cdot]$ means the concatenation of vectors, c_t^e and c_t^d are two attention context vectors described later, and `flag` is what we additionally concatenated to the input embedding, being either 1 or 0,

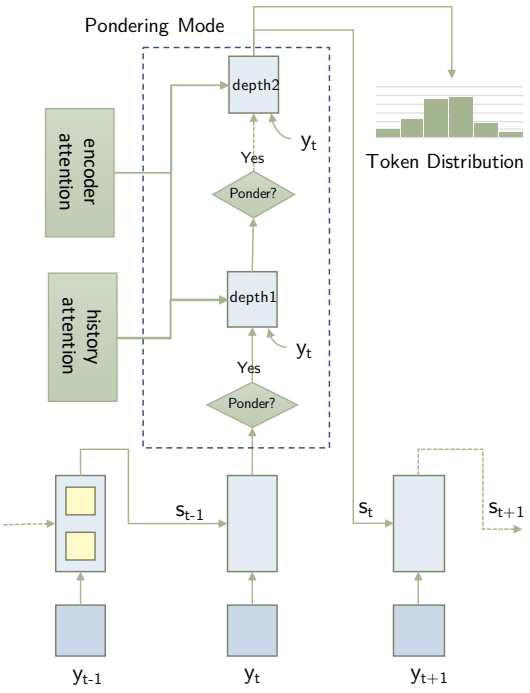

Figure 1: The illustration of our adaptive decoding. Attention and pondering mode are only shown at time t for brevity. Every decoder will go into pondering mode before the next timestep. The decoder cell is a stacked two-layer LSTM and initialized by the last forward states of the corresponeding encoder layer.

based on whether the model is acting in pondering mode or not, which will be introduced later. We further apply a linear mapping and a softmax function to the concatenation of s_t and attention vectors to obtain the word predicting probabilities. We greedily decode the tokens at testing time.

Attention. We adopt two types of attention when decoding. One attends the decoder state upon encoder outputs and yield the input context vector c_t^e,

$$\begin{aligned} \alpha_t^e &= \texttt{Attn}\left(s_{t-1}, \left[h_1^{(2)}, \cdots, h_m^{(2)}\right]\right) \\ c_t^e &= \texttt{Softmax}(\alpha_t^d)\left[s_1, \cdots, s_{t-1}\right] \end{aligned} \quad (2)$$

where $[\cdot,\cdot]$ means to vector stacking. The other similarly attends the hidden state to previous decoder outputs, yielding the context vector c_t^d over the decoding history. We use the bilinear function for encoder attention $\texttt{Attn}(x, y) = x^T \mathbf{W} y + b$, with trained parameters $\mathbf{W}$ and b, and use the dot production function for decoding history attentions $\texttt{Attn}(x, y) = x^T y$.

Halting and Pondering. The key feature of our model is to adaptively choose the decoder depth before predicting tokens. Given the output state s_t

from (1), the model goes into the pondering mode. The output state s_t is further sent to a halting module, which will generate a probability p_t positively correlated with the model uncertainty. We use an MLP with ReLU and sigmoid activations for the halting module. Then a choice is sampled from the Bernoulli distribution determined by p_t. If it chooses to continue, we again use (1) to update the state, meanwhile using the same embedding $\mathbf{y}_t$ for the input.

$$s_t^{(k)} = f_{\texttt{LSTM}}\left(\left[\mathbf{y}_t;\, c_k^e;\, c_k^d;\, \texttt{flag}\right], s_t^{(k-1)}\right) \quad (3)$$

where $s_t^0 = s_t$, $\texttt{flag} = 1$, and c_k^e, c_k^d are attention vectors recomputed with $s_t^{(k-1)}$ using (2). The model will keep pondering until it chooses to stop or reaches our limit of $k = 3$. The final state $s_t^{(k)}$ will act as original s_t in (1) for other modules.

2.2 Uncertainty Estimates

Since the halting module outputs a Bernoulli distribution to guide the decoder, we have to provides some uncertainty quantification for training. Fortunately, Dropout (Srivastava et al., 2014) was proved a good uncertainty estimate (Gal and Ghahramani, 2016). It's simple and effective that neither the model nor the optimization algorithm would need to be changed. We left other estimates like those proposed in Dong et al. (2018) in future work.

To estimate uncertainty with Dropout, we leave the model in training mode and thus the Dropout enabled. We run the forward pass of the equation (3) for F times with the same inputs. Output states are further sent to get token probabilities, $q = \{p(\hat{y}_t = y_{t+1} \mid s_t) \mid \Theta_i\}_{i=1}^F$, where Θ_i is the set of all pertubated parameters affected by Dropout in the i^{th} forward pass. We take the variance of q to reflect model uncertainty $\texttt{U}(s_t) = \texttt{Var}(q)$ as suggested in Gal and Ghahramani (2016). We disable the gradient propagation when computing the variance such that the gradient-based optimization is not influenced.

Note that the variance of a set of probabilities many not be quite large in practice, we thus rescale the variance to make it more numerically robust $\texttt{U}_n(s_t) = \min(\gamma, \texttt{Var}(q))/\gamma$, where $\gamma = 0.15$ in our case.

2.3 Learning

Our model consists of the Seq2Seq part (encoder, decoder, and attention) and the halting module. For the former, we minimize the traditional cross entropy loss with gradient decent, $J^{ent} = \mathbb{E}_{(x,y)} \log p(y \mid x)$.

We use the REINFORCE algorithm to optimize the halting module. The module acts as our policy network, by which the model consecutively make decisions from the action space $\mathcal{A} = \{\texttt{Ponder}, \texttt{Stop}\}$. Each time the model make a choice $a \in \mathcal{A}$, the uncertainty of the seq2seq part is involved in the reward,

$$R(a \mid s_t^{(k)}) = \begin{cases} \texttt{U}_n(s_t^{(k)}) & \text{if incorrect \& } a = 1 \\ 1 - \texttt{U}_n(s_t^{(k)}) & \text{if correct \& } a = 0 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $a = 1$ means a `Ponder` choice and $a = 0$ the other. We measure the correctness by examining the greedily decoded token if $\arg\max_y p(y \mid s_t^k) = y_{t+1}$. The model will be rewarded for a `Stop` action if the prediction is correct, and for a `Ponder` action if the prediction is incorrect. This is similar to the ponder cost of ACT that does not encourage excessive pondering steps.

3 Experiments

We compare our method with other models on two datasets. Our codes could be obtained via https://github.com/zxteloiv/AdaNSP.

3.1 Experimental Setup

Datasets. We use the preprocessed ATIS and GeoQuery datasets kindly provided by Dong and Lapata (2018). All natural language sentences are converted to lower cases and stemmed with NLTK (Bird et al., 2009). Entity mentions like city codes, flight numbers are anonymized using numbered placeholders.

Setups. We choose hyperparameters on the ATIS dataset with the validation set. For the GeoQuery dataset that doesn't come with a validation set, we randomly shuffle the training set and select the top 100 records as the validation set, and the remaining as the new training data. After choosing the best hyperparameters, we resort back to train on the original set. The Dropout ratio is selected from $\{0.5, 0.6, 0.7, 0.8\}$, and the embedding dimension d is chosen from $\{64, 128, 256, 512\}$. We fix the batch size to 20, and both the encoder and decoder cell are two stacked LSTM layers. We apply scheduled sampling (Bengio et al., 2015) with

the ratio 0.2 during training. We run $F = 5$ forward passes before computing the variance. We use Adam (Kingma and Ba, 2015) for the optimizer, and use its default parameters from the paper.

Evaluation. We use the logical form accuracy as the evaluation metric, which is computed with parsed trees of the predictions and true labels. Two trees are considered identical as long as their structures are the same, i.e., the order to sibling predicates doesn't matter. We reuse the STree parser code from Dong and Lapata (2018).

3.2 Results and Analysis

Our model outperforms the other comparative neural semantic parsers on this two set. We reuse the data from Dong and Lapata (2018) since the datasets are identical. Results are listed in Table 1. Our results are better than the SOTAs (Dong and Lapata, 2018; Yin and Neubig, 2018) even without any intermediate representations, whereas Coarse2fine defines a sketch and TranX uses ASDL for every type of target semantic sequences. We outperform Coarse2fine by 0.7% and 0.9% on GeoQuery and ATIS datasets respectively. Although Jia and Liang (2016) has a slightly better result on GeoQuery, they introduced a synchronous CFG to learn new and recombinated examples from the training data, which is a novel method of data augmentation and requires much human effort for preprocessing. For an ablation test, our degenerated model without the pondering part receives considerable performance decreases by 2.8% and 2.9% on GeoQuery and ATIS datasets respectively.

Model	Geo	ATIS
ZC07 (Zettlemoyer and Collins, 2007)	86.1	84.6
λ-WASP (Wong and Mooney, 2007)	86.6	-
FUBL (Kwiatkowski et al., 2011)	88.6	82.8
TISP (Zhao and Huang, 2015)	88.9	84.2
Neural network models		
Seq2Seq (Dong and Lapata, 2016)	84.6	84.2
Seq2Tree (Dong and Lapata, 2016)	87.1	84.6
JL16 (Jia and Liang, 2016)	**89.3**	83.3
TranX (Yin and Neubig, 2018)	88.2	86.2
Coarse2fine (Dong and Lapata, 2018)	88.2	87.7
AdaNSP (ours)	88.9	**88.6**
- halting module	86.1	85.7

Table 1: Results on GeoQuery and ATIS datasets

4 Related Work

Semantic Parsing. CCG or alignment-based Parsers (Zettlemoyer and Collins, 2005, 2007; Kwiatkowski et al., 2010; Wong and Mooney, 2006, 2007) try to model the correlation between semantic tokens and lexical meaning of natural language sentences. Methods based on dependency trees (Ge and Mooney, 2009; Liang et al., 2011; Reddy et al., 2016) otherwise convert outputs from an existing syntactic parser into semantic representations, which can be easily adopted in languages with much fewer resources than English. Recently neural semantic parsers, especially under the encoder-decoder framework, also sprang up (Dong and Lapata, 2016, 2018; Jia and Liang, 2016; Xiao et al., 2016). To make the model aware of the underlying grammar of targets, people try to exert constraints on the decoder side by sketches, typing, grammars and runtime execution guides (Dong and Lapata, 2018; Krishnamurthy et al., 2017; Groschwitz et al., 2018; Wang et al., 2018). Moreover, learning algorithms in SP like structural learning and maximum marginal likelihood are combined with reinforcement algorithms (Guu et al., 2017; Iyyer et al., 2017; Misra et al., 2018).

Adaptive Computing. Adaptive Computation Times (ACT) was first proposed to adaptively learn the depth of RNN models from data (Graves, 2016). Skip-RNN (Campos et al., 2018) used a similar idea to equip a skipping mechanism with existing RNN cells, which adaptively skip some recurrent blocks along the computational graph and thus saved many computations. BlockDrop (Wu et al., 2018) also introduced the REINFORCE algorithm to jointly learn a dropping policy and discard some blocks of the ResNet by the policy network. Recently, Dehghani et al. (2019) proposed Universal Transformers (UT) as an alternative form of the vanilla Transformer (Vaswani et al., 2017). It utilized ACT to control the recurrence times of the basic layer blocks (same parameters) in UT, instead of stacking different block layers in the vanilla Transformer. This helped UT mimic the inductive bias of RNNs and was shown Turing-completed, and has outperformed the vanilla Transformer in many tasks.

5 Conclusion

We present the AdaNSP that adaptively searches the corresponding computation structure of RNNs for semantic parsing. Our method does not need

expert knowledge of intermediate structures of the target sequences, and achieves stronger results than the existing neural semantic parsers.

Acknowledgments

This work is supported by the Natural Science Foundation of China (No.61533018), the Natural Key R&D Program of China (No.2018YFC0830101), the Natural Science Foundation of China (No.61702512) and the independent research project of National Laboratory of Pattern Recognition. This work is also supported by Alibaba Group through Alibaba Innovative Research (AIR) Program, CCF-DiDi BigData Joint Lab and CCF-Tencent Open Research Fund.

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. *Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks*, page 1171–1179. Curran Associates, Inc.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Víctor Campos, Brendan Jou, Xavier Giró i Nieto, Jordi Torres, and Shih-Fu Chang. 2018. Skip RNN: Learning to skip state updates in recurrent neural networks. In *International Conference on Learning Representations*.

Bo Chen, Le Sun, and Xianpei Han. 2018. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1:766–777.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *International Conference on Learning Representations*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 33–43. Association for Computational Linguistics.

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

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1:743–753.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, page 1050–1059.

Ruifang Ge and Raymond J. Mooney. 2009. Learning a compositional semantic parser using an existing syntactic parser. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, page 611–619. Association for Computational Linguistics.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. *arXiv:1603.08983 [cs]*. ArXiv: 1603.08983.

Jonas Groschwitz, Matthias Lindemann, Meaghan Fowlie, Mark Johnson, and Alexander Koller. 2018. Amr dependency parsing with a typed semantic algebra. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 1831–1841. Association for Computational Linguistics.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 1051–1062. Association for Computational Linguistics.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1:1821–1831.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 12–22. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR 2015*. ArXiv: 1412.6980.

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

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language*

processing, page 1223–1233. Association for Computational Linguistics.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, page 1512–1523. Association for Computational Linguistics.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, page 590–599. Association for Computational Linguistics.

Dipendra Misra, Ming-Wei Chang, Xiaodong He, and Wen-tau Yih. 2018. Policy shaping and generalized update equations for semantic parsing from denotations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, page 2442–2452. Association for Computational Linguistics.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming dependency structures to logical forms for semantic parsing. *Transactions of the Association for Computational Linguistics*, 4(0):127–140.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv:1706.03762 [cs]*. ArXiv: 1706.03762.

Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. 2018. Robust text-to-sql generation with execution-guided decoding. *arXiv:1807.03100 [cs]*. ArXiv: 1807.03100.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, page 439–446. Association for Computational Linguistics.

Yuk Wah Wong and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Annual Meeting-Association for computational Linguistics*, volume 45, page 960.

Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, and Rogerio Feris. 2018. Blockdrop: Dynamic inference paths in residual networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. *Proceedings Association For Computational Linguistics, Berlin*.

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. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2018. Tranx: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, page 7–12. 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. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 2:588–594.

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*, page 1653–1663. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, page 658–666. AUAI Press. Event-place: Edinburgh, Scotland.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*, page 678–687.

Kai Zhao and Liang Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1416–1421. Association for Computational Linguistics.