Text-to-SQL Using Seq2Seq Modeling and SQL-Parsing

Jace Mixon
jace.mixon@knights.ucf.edu
University of Central Florida
Orlando, Florida

ABSTRACT

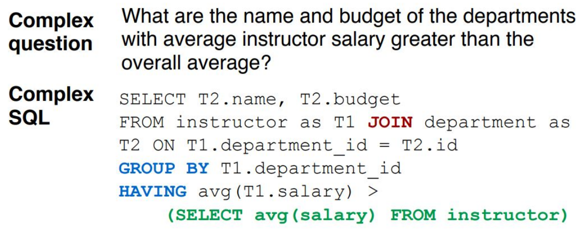
This paper aims to observe the text-to-SQL semantic natural-language problem in machine learning by analyzing the current state-of-the-art approaches for the problem statement: *SmBoP* [11], *NatSQL* [3], and *PICARD* [12]. This also involves incorporating different Seq2Seq models to obtain and examine the results for each approach, which includes the RAT-SQL model and Text-to-Text Transfer Transformer (T5) model [10]. All evaluations of these approaches are done on the Spider 1.0 data set [15], which contains training and testing sets of question-based text inputs and SQL-equivalent outputs that can be used for supervised learning. All results are then calculated using test suites [16] to observe the strengths and weaknesses of each approach in generating the best text-to-SQL model.

ACM Reference Format:
Jace Mixon. 2022. Text-to-SQL Using Seq2Seq Modeling and SQL-Parsing. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A common problem statement within the field of natural-language processing is the semantic parsing task, which involves understanding key elements within an input statement to derive the overall meaning of the statement. This task can usually be found in understanding reviewer comments about an item, such as creating a model to tell if a product review on Amazon was a positive or negative review and then quantifying the proportions of reviews to determine if the product was good or bad based on those reviews. Initially, much of these semantic natural-language processing tasks involved traditional methods of recurrent neural networks, long short-term memory architectures, and gated recurrent neural networks since they have been firmly established as state of the art approaches in sequence modeling. However, in recent years, a shift in model architecture has focused more on an encoder-decoder approach, which attempts to tokenize each individual part of an input sequence that can then be processed by the model efficiently. This is enhanced with different features such as the attention mechanic and multi-head attention that would be the basis of the Transformer model [13] that has outperformed previous state of the art models in semantic parsing.

Conference'17, July 2017, Washington, DC, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn



Complex question What are the name and budget of the departments with average instructor salary greater than the overall average?

Complex SQL

```
SELECT T2.name, T2.budget
FROM instructor as T1 JOIN department as
T2 ON T1.department_id = T2.id
GROUP BY T1.department_id
HAVING avg(T1.salary) >
    (SELECT avg(salary) FROM instructor)
```

Figure 1: Example of a question query as input and sample SQL output from the Spider 1.0 database.

The unique challenge in semantic parsing comes with having to generate tokens rather than simply interpret and classify the input sequence based on a predetermined discrete output. This task can be defined as an interpreter model, where the input is a sentence in one language and the output is the same sentence translated into a different language and it must contain the same semantic meaning in both sentences. In this case, however, this task can also be extended towards generating code. For instance, if we wished to "create a linked list with 5 random integer elements", a semantic parser can interpret the input and generate code in any programming language that can simulate a linked list with 5 integer elements and must be without bugs in the code.

A practical case of creating a semantic parser to generate programming code is in interpreting query statements, such as "how many students are registered for the fall semester?", and generating SQL code that can appropriately query a database and respond with the correct answer, which then means that the SQL code must reflect the semantics of the input question as well. This is known as the "text-to-SQL" task to be accomplished and is the main study point for the research done in this paper. The paper aims to highlight current state-of-the-art approaches to creating a text-to-SQL parser and construct a successful model to perform such task. By the end of the evaluation, we observe the results of each parser- and model-combinations' performance to detail which is the best approach in creating a successful text-to-SQL model, as well as illustrate some of the obstacles in creating such a model.

2 PROBLEM STATEMENT

For the study of this research topic, we want to analyze the text-to-SQL problem statement, which entails that we create a model and pipeline that can translate a natural text-based question into SQL code that can (1) correctly reflect the semantic meaning of the question and (2) contain correct syntax for the SQL code to work in a database schema. A high-scoring model must not only generalize well to currently existing SQL queries but must also to new database schemas.

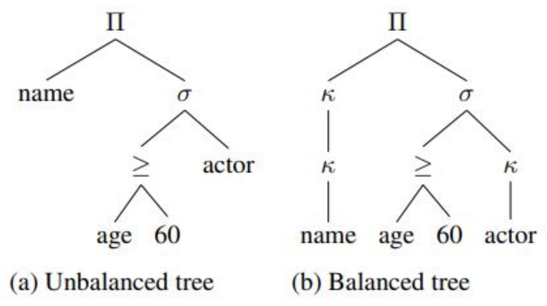
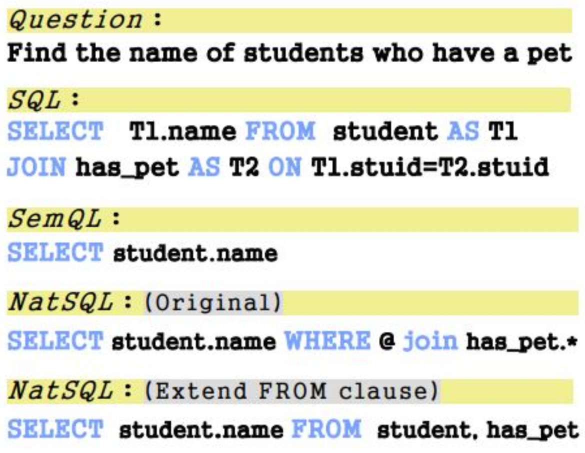


Figure 2: Demonstration of the tree representation in the SmBoP approach to represent the SQL query code.

3 RELATED WORKS

Recent state-of-the-art methods with Seq2Seq architectures have been able to achieves over 80% matching accuracy with text-to-SQL task problems involving complex benchmarks such as ATIS and GeoQuery. This can lead to the discussion that the text-to-SQL problem has been solved and warrants no further discussion on the matter. However, an issue with these approaches have been the predetermined goal of semantic "matching" rather than semantic "parsing", in which the model cannot generalize well onto other database schemas if it does not align with the training data set that the model was using for training. Existing data sets encounter two shortcomings because of this approach. The first problem is that the complex programming [2, 4, 7] are too small in terms of number of programs for training modern data-intensive models and have only a single data set, thus the data set is used for both training and testing of the model and can lead to incorrect evaluations of the model. Additionally, the output label for each of the training and testing sets are limited to a small set of SQL labels that correlate to roughly 4-10 paraphrases. This leads to a situation of models "memorizing" the data set rather than learning from it. The Spider 1.0 database [15] aims to resolve this issue by providing a more extensive data set used for training and testing that are manually annotated and encourage models to generalize to semantic "parsing" rather than semantic "matching" since the data sets are derived from different database schemas.

In regards to the recent upsurge in models generating SQL code from natural-language inputs, they have been studied for a number of years [1, 5, 9], with the more recently successful model, WikiSQL [14], which is the first large-scale cross-domain text-to-SQL data set that has attracted attention from others in the research community. However, it has been denoted in recent research with creating a parser and generator for SQL code that the databases in past years were proven to be insufficient due to the weaknesses described previously. The Spider 1.0 database has been a better alternative for models to use for training and testing and provides better summaries of their performance in generating efficient SQL code for any given database given the Spider 1.0 database providing a variety of test cases that can not be derived from just a singular database schema.



```
Question :
Find the name of students who have a pet

SQL :
SELECT  T1.name FROM  student AS T1
JOIN has_pet AS T2 ON T1.stuid=T2.stuid

SemQL :
SELECT student.name

NatSQL : (Original)
SELECT student.name WHERE @ join has_pet.*

NatSQL : (Extend FROM clause)
SELECT  student.name FROM  student, has_pet
```

Figure 3: Example outputs for the NatSQL parser that simplifies the output SQL code in comparison to the target label SQL code.

4 TECHNIQUE

For evaluation of the best model that can generate SQL code based on a natural-language question input, a combination of the selection of database that contain the natural-language question inputs and matching SQL code output, Seq2Seq recurrent neural network architecture, and semantic SQL parser will be used and specifically chosen to best fit each other to provide the best result possible. Some of the selections will cause restrictions of other selections in the combination due to the authors of the models or parsers' design, which will be obeyed and will be discussed in a later section.

4.1 Database

The problem statement is unique in that it does not generate natural-language output based on natural-language input, as mentioned previously. Therefore, it requires a specialized database to use for training that formats natural-language questions as input and matches to SQL queries as outputs. For such a task, we opted to use Spider 1.0 [15], a Yale Semantic Parsing and Text-to-SQL database that is designed for training models to solve the problem statement. The Spider 1.0 database is a large-scale complex and cross-domain database that is manually annotated and consists of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains.

4.2 Model Selection

With the problem statement being natural-language processing problem statement, it makes sense to select a recurrent neural network model or a model that consists of long short-term memory modules or gated recurrent units since they are a popular choice for natural-language processing. However, for the purpose of this research topic, we opted to use a more powerful model that can handle a complex problem statement. Therefore, we opted to use a sequence-to-sequence (Seq2Seq) model, which incorporates an encoder-decoder structure with the help of Attention [13] that can appropriately process the complex question inputs and generate

SQL code rather than make text prediction on the natural language vocabulary.

The specific selection of Seq2Seq model will vary, such as using the provided baseline Seq2Seq model in the Spider 1.0 database and using other contemporary models such as the Text-to-Text Transfer Transformer (T5) model [10]. While the model will play a major role in the overall performance to solve the problem statement, the additional key factor to help boost its performance is with the parser, in which there are three that will be explored.

4.3 SQL Parser

Since the primary focus of the results and discussion will be centered around the parsers and their overall effectiveness in solving the problem statement, there are three parsers that will be examined:

4.3.1 *SmBoP.*

Semi-autoregressive Bottom-up Semantic Parsing [11], or SmBoP, focuses on the integration of a unique beam search technique that involve storing high-scoring trees that contain schema constants or DB values with a SQL encoder-decoder model (RAT-SQL) that provides joint contextualized representation of utterances and schema. The RAT-SQL model also uses relational-aware self-attention that encodes structures of the schema based on relations between encoded tokens.

While this approach specializes solely on the text-to-SQL problem statement and therefore allows for possibly the best performance out of the listed parsers, the issue arises in its generality, meaning that it is limited to a singular model and therefore prevented from being generalized to other Seq2Seq models.

4.3.2 *NatSQL.*

Natural SQL [3], or NatSQL, recognizes the complexity of the resulting SQL queries generated by other encoder-decoder models and aim to create a parser that generates the syntactically correct query code while also retaining the semantic meaning of the code in its relation to the input question. It aims to accomplish this task by removing and replacing certain keywords and operators in SQL and therefore condense the query code into a compact statement.

This can lead to overall speedup in performance since it can generalize to alternative schemas and database without sacrificing accuracy due to the generality the parser achieves. However, we aim to see the accuracy of the parser and observe it increase or decrease since resulting SQL queries that are simplified can lose accuracy due to loss in generality.

4.3.3 *PICARD.*

Parsing Incrementally for Constrained Auto-Regressive Decoding [12], or PICARD, performs by finding valid output sequences by rejecting inadmissible tokens at each decoding step in the decoder. Based on the author's claims, this parser allows for text-to-SQL models to transform from having passable performance to state-of-the-art solutions.

During the writing of this report, it has been highlighted that PICARD has obtained the best resulting model for the text-to-SQL problem statement, so our preliminary expectations will be that PICARD will outperform the other two parsers listed previously.

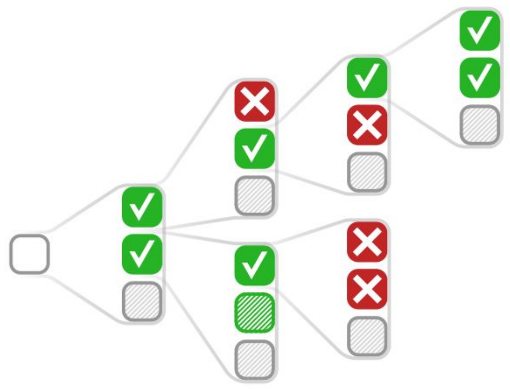


Figure 4: Demonstration of the restricted beam search done by PICARD. Each column represents three token predictions with the highest probability ordered at the top. The green check marks indicate an accepted input while the red crosses indicate a rejected input, based on the PICARD algorithm.

5 EVALUATION

All semantic parsers and models were trained with their own set of parameters to ensure the best possible output, which will be described in each results' section. The constant comparison between each semantic parser approach is the selection of test suites used to evaluate each methodology. It is to be noted that the authors for the Spider 1.0 data set do contain secrete test suites that are used in evaluating the effectiveness of the model and parser, which leads to the results that they have generated and it is to be expected that they will receive different performance ratings than the ones shown in this paper since this current evaluation is done with publicly-available test suites that all researchers have access to when creating their approach.

For each result, the execution accuracy and exact matching accuracy will be analyzed. Execution accuracy details the accuracy of the generated SQL code from the model and parser combination, which can be shown to be higher than the matching accuracy because multiple SQL code can represent the same natural-language question that it attempts to represent. The importance for execution accuracy is that the same results match with the expected outcome of the natural-language question. The exact matching accuracy is an additive measurement to indicate how well the model and parser's SQL code output matches the test suite's provided output SQL code. Obtaining a large accuracy in this measurement is not as important as the execution accuracy since the important objective for each approach is to create a generalized approach that can be generalized to any database schema instead of adapting to a singular database schema.

The last detail to mention is that all training and testing were done on an Nvidia RTX 2070 Super, which may differ from the training done for each approach's papers and may incur a heavy overhead to the inference time, but the inference and training time length will not be analyzed in this section.

	Exact Match Accuracy	Execution Accuracy
SmBoP	52.3%	66.3%
NatSQL	90.5%	91.7%
PICARD	58.1%	63.5%

Table 1: Comparison results of all mentioned parser methods.

5.1 SmBoP Results

The SmBoP approach uses a modification of a model known as GRAPPA, which consists of 24 Transformer layers followed by another 8 RAT-SQL layers, which are all implemented inside AllenNLP [6]. The beam size is K = 30 and the number of decoding steps is T = 9. For the tree representation, it uses a transformer of one layer, 8 heads, and dimensionality of 256. It is trained for 60K steps with batch size 60 and early stopping is used once convergence has been reached.

Upon training the model and utilizing the parser, SmBoP achieves an execution accuracy of 66.3% accuracy and an exact matching accuracy of 52.3%. This differs from the paper's observation of the methodology by just under 20% since the author's observations were that the exact matching accuracy was 69.5% and the execution accuracy was 71.1%.

5.2 NatSQL Results

For the NatSQL approach, the authors used RAT-SQL as their basis model, which is similar to the SmBoP approach except that SmBoP uses modified portions of RAT-SQL to compose their model. For this approach, the model's training parameters mimics similarly to the training in SmBoP. It is especially important to use RAT-SQL in this replication of the experiment since it was noted in the NatSQL paper that different model combinations with this parser caused a performance decrease, with RAT-SQL providing the best result possible for this parser.

The author's evaluation of the parser and model observes an exact matching accuracy of 73.7% and an execution accuracy of 75.0%. Surprisingly, when replicating the experiment, results showed a 90.5% accuracy in exact matching and a 91.7% in execution matching. The results in the replicated experiment have tremendously exceeded the results in the paper. Further discussion of this possible outcome will be explained in a later section.

5.3 PICARD Results

In the PICARD paper, the authors primarily used the T5 transformer model and they varied the different types of T5 transformer models in their results. For the replication of this experiment, the T5-3B was selected to be used as the model for the PICARD parser. For the PICARD analysis, the authors fine-tuned the T5 model for up to 3072 epochs using Adafactor [8], a batch size of 2048, and a learning rate of 10^{-4}. This could not be achieved in the replication of the experiment due to the vast overhead of the required resources it would demand on the hardware. This leads to the replication of the experiment to use a fine-tuned model of the T5 that has been trained for only 10 epochs and a batch size of 100.

For the results of PICARD, the authors have demonstrated findings of 71.9% in the exact matching accuracy and a 75.1% in the execution accuracy, which has been the leading scores for the text-to-SQL problem statement in the Spider 1.0 database. Upon replicating the experiment, the results that were observed was 58.1% in the exact matching accuracy and a 63.5% in the execution accuracy of the model and parser approach.

6 DISCUSSION

When initially replicating the experiments done from each parser, PICARD was the first parser selected for testing. This method was selected initially due to the parser being malleable to any given Seq2Seq model since it does not impact the model directly and can provide result improvements. However, upon reflection of the experiments, it was the NatSQL approach that appeared to have the best flexibility for implementation upon any model and hardware, whereas PICARD and SmBoP showed more complications in replicating the experiments, which could be due to the massive hardware constraints to satisfy the algorithm and, for SmBoP, the algorithm was tailored towards the GRAPPA model initially and thus the provided code was with the model in mind.

Regardless of the obstacles for each implementation of Seq2Seq model and parser, each replication of the experiments showed results that were unexpected in comparison to initial impressions. The best performing parser was the NatSQL in both the exact match accuracy and execution accuracy, whereas the lowest score for exact match accuracy was with SmBoP at a 52.3% accuracy and the the lowest execution accuracy score was with PICARD at a 63.5%. This is an unusual observation with NatSQL because the objective with NatSQL is that it generates an SQL code that is a simplified version of the target SQL code for the test suites to provide better generalization to other database schemas, therefore the exact-match accuracy should be drastically lower since the NatSQL algorithm aims to create a different SQL query syntactically but represents the same input question semantically. However, this contradicts that idea with an impressive score of 90.5% and outperforming all other methodologies. The inconsistencies can be explained from two different angles. The first perspective is that the test suite's exact match accuracy could have a different way of calculating the measurement than simply examining the syntactical similarities between the target code and output code through examining each token individually. The second perspective is that after the NatSQL paper has been published, the authors continued to improve the source code, which then led to the discovery of a better algorithm to replicate than what was initially proposed.

The SmBoP approached faced numerous obstacles in implementation for the experiment replication. This is in part because of how limited the algorithm is to translate to different models, therefore the configuration for the SmBoP parser was severely limited to what the authors have proposed in their research. Possible future work for this parser could be to extend their findings to more robust Seq2Seq models, such as the T5 model used by PICARD. It could also be understood that the authors have attempted to use different models to implement their parser and the GRAPPA modified model produced the best output possible, but the paper never explicitly states if this was the case. Additionally, the use of tree representations for the SQL code is known to be computationally expensive, which can cause severe performance overhead, which is not an optimal situation for the approach to have.

The most interesting observation made in the experiment replication is the PICARD replication. The PICARD method was highlighted in the Spider 1.0 database to have the highest score among all methodologies proposed on the Spider 1.0 database. Yet, the PICARD algorithm performed sub-optimally in comparison to NatSQL and did not achieve the same score as the paper and Spider 1.0 database stated, but it could be due to a number of reasons. The most obvious reason for the difference in score could be due to the changes in training parameter when fine-tuning the T5 architecture, in which the parameters were drastically reduced to perform the experimentation and gather the results for the PICARD parser. Additionally, much like with the SmBoP approach, even with the reduced parameter settings introduced for the fine-tuning process, the algorithm incurred a huge amount of performance overhead to account for, which drastically slowed down the process for PICARD to process the inputs and generate the SQL code output.

7 LIMITATIONS

Much of the challenges with the research topic has been creating a platform that can uniformly examine the strengths and weaknesses of the parser and model effectiveness. As demonstrated in the results for each parser, the SmBoP approach was fully designed with the GRAPPA model in mind and the PICARD method, albeit still works on any given model, was tested and analyzed using the T5 architecture. The NatSQL was the only portable method that could work within the hardware restraint provided for replicating the experiment results, but the paper for NatSQL did demonstrate their results using the RAT-SQL model, which was then replicated to produce the best result possible. This could lead to the perspective that each parser was given the best model to work with to produce the best result possible and, in turn, is a biased way to perform the experiment, but the choice to select the best model for each parser was done due to the restraint that SmBoP presented by itself. Regardless, each model and parser has been developed independently of each other, and therefore they are designed with different pipelines and structures which can cause conflict in how they are interpreted during testing.

Much of replicating these experiments were also difficult in implementing onto the current hardware for this paper since the hardware restriction was based on availability of the GPU, whereas the authors within their research had access to more powerful hardware that can support the training and testing pipelines that they have constructed to offer the best results possible for their methodology. An example of these complications is in implementing PICARD and through the use of Docker, which the process itself consumed close to 30 GB worth of storage space alone, and the rest of the model and parser algorithm occupied close to 100 GB. Additionally, the parameters set for training the algorithm is set for intensive training procedures, which is evident by the 3072 epochs and 2048 batch size when fine-tuning the T5 model.

The performance overhead also impacted the inference timing of the models and parser when analyzing each individual sample. For instance, the PICARD paper observed that for inference time on an Nvidia A100-SXM4-40GB GPU averaged 2.5 seconds per sample without PICARD and 3.1 seconds per sample with PICARD, whereas on the hardware for the experiment replication, the inference time could not be measured since each sample processed, with and without PICARD, required several seconds to the point where there was a concern that the program would not halt.

The main takeaway from the limitations observed in this paper is that while the results of the parsers seem to either exceed the paper's results or were less than the paper's results, the results could be heavily influenced by the limitations presented in the hardware and the changes in parameter settings to reduce the overhead in training and testing the models.

8 CONCLUSION

This paper aimed to observe the different parser methods and models for the text-to-SQL semantic task in natural-language processing. The different parsers under observation were the Semi-autoregressive Bottom-up Semantic Parsing (SmBoP) algorithm, the Natural SQL (NatSQL) algorithm, and the Parsing Incrementally for Constrained Auto-Regressive Decoding (PICARD) algorithm. The models under observation were the Text-to-Text Transfer Transformer (T5) model and the RAT-SQL model, which featured a variation of the model called GRAPPA model, which was used solely in the SmBoP parser. By replicating the experiments conducted for each parser and using the model that guaranteed the best results possible from each parser, all semantic parsers were trained using the Spider 1.0 database and tested using the test suites to provide a quantifiable accuracy measurement between each parser. The NatSQL parser demonstrated the best resulting parser and model combination, which was different than the preliminary impressions of the NatSQL parser, especially in comparison to the PICARD algorithm. While the results are comparable, it is noted that the hardware provided in this paper could not replicate all experiments equally and, therefore, future work can be done by using better hardware that can satisfy all experiment parameters to provide a justifiable measurement accuracy.

REFERENCES

[1] Fernando C N Pereira David H D Warren. 1982. An Efficient Easily Adabtable System for Interpreting Natural Language Queries. *Comput. Linguist* 8, 3-4 (1982), 110–122.

[2] HV Jagadish Fei Li. 2014. Constructing an interactive natural language interface for relational databases. *VLDB* (2014).

[3] Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. 2021. Natural SQL: Making SQL Easier to Infer from Natural Language Specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics, Punta Cana, Dominican Republic, 2030–2042. https://doi.org/10.18653/v1/2021.findings-emnlp.174

[4] Raymond J. Mooney John M. Zelle. 1996. Learning to parse database queries using inductive logic programming. *AAAI/IAAI* (1996), 1050–1055.

[5] Mirella Lapata Li Dong. 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)*. 731–742.

[6] Mark Neumann Oyvind Tafjord Pradeep Dasigi-Nelson F. Liu Matthew Peters Michael Schmitz Luke Zettlemoyer Matt Gardner, Joel Grus. 2018. AllenNLP: A dep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. 1–6.

[7] Isil Dillig Thomas Dillig Navid Yaghmazadeh, Yuepeng Wang. 2017. Sqlizer: query synthesis from natural language. In *Proceedings of the ACM on Programming Languages*.

[8] Mitchell Stern Noam Shazeer. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*. 4596–4604.

[9] Elena-Simona Apostol Ciprian-Octavian Truica Ionel Alexandru Hosu Traian Rebedea Radu Cristian Alexandru Iacob, Florin Brad. 2020. Neural approaches for natural language interfaces to databases: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*. 381–395.

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67. http://jmlr.org/papers/v21/20-074.html

[11] Ohad Rubin and Jonathan Berant. 2020. SmBoP: Semi-autoregressive Bottom-up Semantic Parsing. https://doi.org/10.48550/ARXIV.2010.12412

[12] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 9895–9901. https://aclanthology.org/2021.emnlp-main.779

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. https://doi.org/10.48550/ARXIV.1706.03762

[14] Richard Socher Victor Zhong, Caiming Xiong. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR* (2017).

[15] 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. 2018. 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*. Association for Computational Linguistics, Brussels, Belgium.

[16] Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic Evaluation for Text-to-SQL with Distilled Test Suite. In *The 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.