A Deep Dive into Deep Learning Approaches for Text-to-SQL Systems

George Katsogiannis-Meimarakis
katso@athenarc.gr
Athena Research Center
Athens, Greece

Georgia Koutrika
georgia@athenarc.gr
Athena Research Center
Athens, Greece

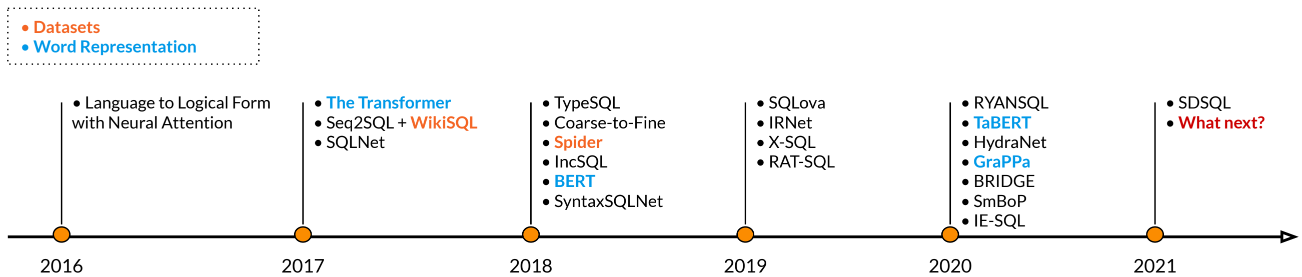

Figure 1: Deep Learning Text-to-SQL Timeline

ABSTRACT

Data is a prevalent part of every business and scientific domain, but its explosive volume and increasing complexity make data querying challenging even for experts. For this reason, numerous text-to-SQL systems have been developed that enable querying relational databases using natural language. The recent advances on deep neural networks along with the creation of two large datasets specifically made for training text-to-SQL systems, have paved the path for a novel and very promising research area. The purpose of this tutorial is a deep dive into this area, covering state-of-the-art techniques for natural language representation in neural networks, benchmarks that sparked research and competition, recent text-to-SQL systems using deep learning techniques, as well as open problems and research opportunities.

CCS CONCEPTS

• **Information systems** → **Search interfaces**; • **Computing methodologies** → **Natural language processing**; **Neural networks**.

KEYWORDS

Text-to-SQL, Deep Learning

ACM Reference Format:
George Katsogiannis-Meimarakis and Georgia Koutrika. 2021. A Deep Dive into Deep Learning Approaches for Text-to-SQL Systems. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD ’21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3448016.3457543

ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3457543

1 INTRODUCTION

Relational databases hold a vast amount of data, necessary in a wide range of tasks, from business operations, medical and scientific research to activities in our everyday lives. However, they remain inaccessible to non-technical users without knowledge of SQL and the underlying database schema. During the past decades, there has been an increasing research focus on natural language interfaces that can lift this barrier [1, 3, 22]. Additionally, the recent advances on deep neural networks along with the creation of two large datasets made for training text-to-SQL systems, have paved the path for a very exciting, but at the same time highly competitive and fast-paced, research field. Text-to-SQL systems based on deep learning are popping up “like mushrooms after a rain” aiming at providing better solutions to the notoriously hard text-to-SQL problem. A systematic study of these solutions is missing.

We believe that in order to make real progress in building text-to-SQL systems, we need to de-mystify what has been done, understand how and when each model and approach can be used, and, finally, distinguish the research challenges ahead of us. Therefore, in this tutorial, we follow a systematic and structured approach. First, we introduce the audience to the text-to-SQL problem, explain and categorize its challenges based on their source and complexity. Then, we present available benchmarks and explain their advantages and shortcomings. We zoom in on the recent advances of deep learning techniques for text-to-SQL translation. We organize them in a detailed taxonomy and highlight their differences and commonalities as well as their advantages and deficiencies. Our analysis will highlight new research opportunities for researchers and practitioners in the fields of database systems, natural language processing and deep learning.

The objectives of our tutorial are three-fold:

- Through our in-depth survey of deep learning approaches, we aim at building a strong foundation for ongoing research and a research agenda for text-to-SQL systems going forward;
- While natural language interfaces have been the holy grail of the database community and notable systems have been proposed [6, 16, 28, 30, 34], as a community we are being surpassed by the machine learning and NLP communities, which are turning their attention to this problem. With this tutorial, we aim at catching up and inspiring new solutions that combine the best of the database, ML, and NLP worlds.
- Going beyond the text-to-SQL problem, our tutorial aims at equipping the audience with ideas and methods to transfer over to solutions for other database problems, including NL explanations, query recommendations, data exploration, information extraction and integration.

2 TUTORIAL OUTLINE

2.1 The Text-to-SQL Problem

We will introduce the problem at hand, give a complete definition, present its main challenges, and analyze their impact on a text-to-SQL system. The text-to-SQL (also known as NL2SQL) problem can be described as follows: *Given a Natural Language Query (NLQ) on a Relational Database (RDB), produce a SQL query equivalent to the NLQ, which is valid for the said RDB.* Despite the simplicity of this definition, the complexity and difficulty of the task should not be underestimated.

Ambiguity of natural language queries is one of the most difficult challenges the systems have to solve. There are several types of ambiguity [2, 23]. For instance, *lexical ambiguity*, where a single word has multiple meanings (e.g., *"Paris"* can be a city or a person), and *syntactic ambiguity*, where a sentence has multiple interpretations (e.g., *"Find all German movie directors"* can mean *"directors that have directed German movies"* or *"directors from Germany"*).

Schema linking is the problem of understanding which parts of the NLQ refer to which parts of the database schema. Occasionally, this can be solved by performing text matching but this is not always the case. Another, related, concern is the *vocabulary gap*, i.e. the differences between the vocabulary used by the database and the one used by the user. *User mistakes*, such as syntactical or grammatical errors, make the problem even more challenging.

Answer Validation is very important. Since in many cases users are not be familiar with SQL, the question is how they can confirm that the obtained results match the intention of the NLQ. This problem becomes all the more relevant when, for example, the user is conducting medical research and is querying a database that contains medical data. In this case, producing an incorrect translation could be catastrophic.

Another aspect of the problem that must be kept in mind is the *universality of the solution*, i.e. the system's ability to perform equally well on different databases. Ideally, a text-to-SQL system should be able to handle different domains and schemas with little assistance or effort from a human maintainer. It is also important to enable natural language queries in languages other than English.

2.2 Available Benchmarks

We will introduce the benchmarks that are available to the research community for developing a text-to-SQL system. Training a deep learning system is a very data-intensive procedure; large amounts of data are required in order to train an accurate model. For this reason, the availability of datasets is the main fuel for the development of deep learning solutions and the text-to-SQL task is no exception. This becomes even more obvious considering that the bloom of deep learning research for the text-to-SQL problem starts when the first large dataset is released [46]. We will briefly mention previous small-scale benchmarks, such as ATIS [26], Scholar [15], IMDB and YELP [37], and then present WikiSQL and Spider, the largest and most popular benchmarks for the text-to-SQL problem.

WikiSQL [46] is *a large crowd-sourced dataset for developing natural language interfaces for relational databases* released along with the Seq2SQL text-to-SQL system. It contains over 25,000 tables gathered from Wikipedia pages and over 80,000 natural language and SQL question pairs, which were created by crowd-sourcing. Note that each of WikiSQL's questions is directed to a single table and not to a relational database. This means that the proposed task is much simpler than the ultimate goal of creating a natural language interface for relational databases. Additionally, the complexity of the queries is very low. There are no JOIN, GROUP BY, UNION, INTERSECTION or other complex SQL elements. We must also note that that WikiSQL contains multiple errors and ambiguities, which might hinder the performance of any model trained on it. Research even suggests that the state-of-the-art systems have reached the upper barrier of accuracy on the task [14].

The Spider dataset [44] *is a large-scale complex and cross-domain semantic parsing and text-to-SQL dataset* annotated by 11 Yale students. It contains 200 relational databases from 138 different domains along with over 10,000 natural language questions and over 5,000 complex SQL queries. Its queries cover a wide range of complexity, from very simple to very hard, using all the common SQL elements and including nesting. All the above, along with the fact that it was hand-crafted and re-checked are an indicator of its quality and of the fact that it can support the development of very promising systems.

2.3 NL Representation

We will provide an overview of the state-of-the-art techniques for natural language representation in neural networks, as well as some insights about open research paths on the matter.

The text-to-SQL demands state-of-the-art NL processing techniques if the proposed solution is to be efficient. Firstly, the use of neural networks, which can only handle numerical inputs and not raw text, has led to the use of word embeddings for numerical word representation. Additionally, in the past few years, the use of language models is blooming, following their rise as an efficient solution for increased performance in NL tasks.

Word embeddings assume that every unique word has a numerical representation that can be different from all other words and at the same time incorporate useful information about the word, and aim at mapping each word to a multidimensional vector. Besides the brute-force creation of one-hot embeddings, researchers have provided highly efficient techniques to create representations that carry the word's meaning and its relationships with other words. Word2Vec [21], GloVe [25] and WordPiece embeddings [35], to name a few, are some famous word embedding techniques that are used in most, if not all, text-to-sql systems.

Language models are a novel and emerging type of pre-trained neural networks for processing NL, that has been shown to excel in NL tasks during the past few years. Note that language models are not a replacement for word embeddings, given that they are neural networks and they still need a way of transforming words to vectors. The way this type of models are created, is that a very large network (10^8 order of magnitude of parameters) is created and is pre-trained on a very large NL dataset (10^9 order of magnitude of words). The pre-trained model is made available for researchers who can then adapt its inputs and outputs, to the specific task they aim to solve, and train it for an additional number of epochs on their task-specific dataset. The result is a much stronger model that can reach state-of-the-art performance even without the need of complex architectures [5]. These models have been able to reach such performances due to the use of a neural network architecture that was recently proposed, called the Transformer [31], which excels at handling NL and sequences of NL where the connections between the words are important. Some of the most used language models for the text-to-SQL task are BERT [5] and MT-DNN [18], while new models pre-trained specifically for tasks handling structured data are emerging such as TaBERT [39] and GraPPa [41], which have shown to be quite promising.

2.4 Text-to-SQL Deep Learning Approaches

We will give a taxonomy of deep learning approaches for text-to-SQL, and highlight the main characteristics as well as the advantages and shortcomings of each neural network architecture. Deep learning systems following the encoder-decoder architecture can be distinguished in three categories, based on the output of their decoder [4]: (a) sequence-to-sequence approaches, (b) grammar-based approaches, and (c) sketch-based slot-filling approaches.

Sequence-to-sequence approaches produce a sequence of SQL tokens and schema elements as their output, with the resulting sequence being the final SQL query prediction, or a major part of it. Essentially, they attempt to transform an input NLQ sequence to an output SQL sequence. This approach is the simplest, but is also very prone to errors. It was adopted by one of the first deep-learning systems for the task at hand, Seq2SQL [46], but later systems steered away from such approaches. The main drawback of sequence-to-sequence architectures is that they do not take the strict grammatical rules of SQL into account when generating a query. The system attempts to learn how a SQL sequence is generated, but at prediction time there are no measures to safeguard from producing syntactically incorrect queries. It must be noted however that this does not necessarily mean that there are no later efforts in developing seq-to-seq systems. For example, BRIDGE [17] uses a seq-to-seq architecture while leveraging the power of BERT [5] as well as *schema-consistency guided decoding* to avoid the aforementioned drawbacks.

Grammar-based approaches are an evolution of sequence-to-sequence approaches, and produce a sequence of grammar rules instead of simple tokens as their output. These grammar rules are instructions that, when applied, can create a SQL query. The advantage over sequence-to-sequence approaches is that the possibility for generating an out-of-place token or a syntactically incorrect query is dramatically reduced. This is the most used approach for generating complex SQL queries and has been adopted by many systems such as RAT-SQL [32], IRNet [10], IncSQL [29], RYANSQL [4], Coarse-to-Fine [8], SyntaxSQLNet [42] and SmBoP [27].

Sketch-based slot-filling approaches aim at simplifying the difficult task of generating a SQL query to the easier task of predicting certain parts of the query (e.g. which of the table columns will appear in the SELECT clause), transforming in this way the SQL generation task to a classification task. In this case, we consider a query sketch with a number of empty slots that must be filled and develop neural networks that predict which element is most probable to fill each slot. A basic prerequisite for such approaches is to have a query sketch that, when filled, will be able to capture the NLQ's intention. As a result, this category of systems is rarely able to produce complex SQL queries. Some examples of sketch-based text-to-SQL systems are SQLNet [36], HydraNet [19], SQLova [14], X-SQL [11], IE-SQL [20], RoBERTa-NL2SQL [24] and TypeSQL [40].

2.5 Key Text-to-SQL Systems

Multiple text-to-SQL systems will be studied in greater depth to offer a concrete understanding of how each system tackles the problem and the range of techniques that have been proposed. We will also take a look at techniques that have been widely adopted by text-to-SQL systems to improve the quality of their predictions, such as Execution-guided Decoding [33].

Besides specific approaches proposed by each system, there are some common elements that are found in most systems. For example, since a sequence of words must be processed, it is very common to see recurrent neural networks such as the LSTM [12] or attention-based networks such as the Transformer [31], in recent systems. Additionally, all networks use word embeddings, while most recent systems incorporate some language model, as well. Below, we zoom in on example milestone systems.

Seq2SQL [46] was one of the first neural networks created specifically for the text-to-SQL task and was based on a previous work focusing on generating logical forms using neural networks [7]. Its authors released the WikiSQL dataset along with it, which signified a new era for deep learning research on the text-to-SQL problem. The system predicts an aggregation function and the column for the SELECT clause as classification tasks and generates the WHERE condition clause using a seq-to-seq network. The latter part of the system is burdened with generating parts of the query that can lead to syntactic errors, which is its major drawback. The network architecture combines LSTM and linear layers, and the GloVe embeddings are used to represent the inputs.

SQLNet [36] was one of the first sketch-based approaches. It was based on the observation that the way Seq2SQL chose to generate the WHERE clause was prone to errors that could be avoided. For this reason, a query sketch, that could cover every SQL query in the WikiSQL dataset, was developed and separate neural networks were created to fill each slot. All slots are filled by considering a classification task (e.g., which of the six possible aggregation functions is appropriate for the given NLQ) except for the condition value slot which was generated by a seq-to-seq network. Note that in this case the aforementioned seq-to-seq network only generates a value and does not handle SQL tokens, meaning that it is not possible to generate syntactically incorrect queries. Another improvement is the introduction of a *column attention neural architecture* to the network. Other than that, the network resembles Seq2SQL, using a combination of LSTM and linear layers.

HydraNet [19] focuses on the WikiSQL task and follows a sketch-based approach, using the same sketch as SQLNet, but takes advantage of the BERT language model.

SQLova [14] is another sketch-based approach focusing on the WikiSQL dataset and leveraging the BERT language model, just as the HydraNet system. Their main difference is that while HydraNet aims to use a very simple network after receiving BERT's output, SQLova employs a large and complex network similar to the one used by SQLNet, while also incorporating BERT into the system. What must be noted is that even though SQLova employs a larger and more complex network than HydraNet, it achieves lower accuracy scores on the WikiSQL dataset.

BRIDGE [17] is a rare example of a recent seq-to-seq based approach. Besides incorporating BERT for better NL processing and using fuzzy string matching for schema linking, it also introduces schema-consistency guided decoding to avoid errors at prediction time. For example, it forces the system to predict SQL keywords in a specific order and prevents it from generating any column names if it has not already generated the name of the table they appear in.

SDSQL [13] is one of the latest sketch-based systems to be proposed. It uses Schema Dependency Learning to tackle the schema linking problem, essentially teaching the system to find connections between NLQ tokens and table columns. This is an interesting approach, compared to systems that perform schema linking using text matching methods (e.g. RAT-SQL, IRNet).

RAT-SQL [32] is a grammar-based text-to-SQL system focusing on the Spider dataset. It is capable of generating complex SQL queries by incorporating three note-worthy features. First, it creates a *question-contextualized schema graph*, i.e. a graph representing the database schema, its tables and columns, as well as the words of the user's question as nodes and the connections between them as edges. The edges between DB elements are created based on the DB schema and the edges between NLQ words and DB elements are created by performing text matching, which is a form of schema linking. Furthermore, it uses a modified Transformer network for *relation aware self-attention*, that is specifically designed to leverage the information of the created graph and its edges. Finally, it follows a method for SQL generation as an abstract syntax tree, by generating a sequence of actions for building the tree, as proposed in [38].

IRNet [10] is another grammar-based system capable of generating complex SQL queries. It uses text-matching techniques to address the schema linking challenge similarly but in a simpler form than RAT-SQL. It uses a complex architecture of linear and recurrent neural networks to process the input, in addition to BERT. After processing the input, it creates an SQL query using the same method as RAT-SQL for generating an abstract syntax tree, with the main difference that the output it produces is in a intermediate language called SemQL designed specifically for this system. Its authors argue that it is easier to generate queries in this language and then transform them to SQL.

2.6 Challenges and Research Opportunities

We will discuss about the need for new benchmarks and in-depth system evaluations [9] and how the database community can help complement the work done by benchmarks such as Spider. While the state-of-the-art systems are still dealing with 'getting the answer right', they are mostly overlooking the 'getting the answer fast'. Hence, the database community could come up with benchmarks that focus on efficiency (instead of effectiveness), and allow evaluating systems based on execution time and resource consumption.

We will discuss how far existing systems have gone and outline pressing issues that need to be tackled. Queries with synonyms, misspellings, negation, as well as queries with more complicated logic are open challenges. We need to work on them since users are prone to asking queries that exhibit such issues.

We will discuss how database, ML, and NLP approaches can join forces to push the barrier further. Building text-to-SQL systems in the intersection of these domains promises to combine the best of these worlds. In practice, it also raises several challenges. For instance, how should a system combine components that use different approaches? Which approach works best for which problem?

Developing a conversational DB interface is another promising task, very similar to earlier non-DL approaches such as Analyza [6], which heavily involves the user in the translation process. The recent release of a conversational text-to-SQL dataset (CoSQL [43]), will surely bring additional focus on the task, while also enabling deep learning research. The same goes for the context-dependent alternation of the text-to-SQL task and an equivalent dataset which was recently made available (SParC [45]).

Generalizing the text-to-SQL problem to other querying languages and data storage options is another domain that will also enjoy the attention of researchers in the near future. Even though the relational model is the most common approach for storing data, the advancements of ontologies, the Resource Description Framework (RDF) and query languages such as SPARQL, point to the need for similar interfaces that can generalize beyond SQL.

Finally, we will look into other data-related problems, including NL explanations, query recommendations, data exploration, information extraction and integration, and discuss how ideas and methods from the text-to-SQL problem could be transferred over to inspire new solutions for these problems.

3 PRESENTERS

George Katsogiannis-Meimarakis is a research assistant at Athena Research Center in Greece, where he works on the INODE (Intelligent Open Data Exploration) project, focusing on the text-to-SQL problem. He is a graduate of the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens, where he completed his thesis with the title "Translating Natural Language to SQL using Deep Learning". Currently, he is attending a MSc programme on Data Science and Information

Technologies with a specialisation on Artificial Intelligence and Big Data.

Georgia Koutrika is a Research Director at Athena Research Center in Greece. She has more than 15 years of experience in multiple roles at HP Labs, IBM Almaden, and Stanford. Her work focuses on data exploration, recommendations, and data analytics, and has been incorporated in commercial products, described in 14 granted patents and 26 patent applications in the US and worldwide, and published in more than 90 papers in top-tier conferences and journals. She is Editor-in-chief for VLDB Journal, PC chair for VLDB 2023, associate editor for TKDE, and an ACM Distinguished Speaker. *Prior tutorials:* Fairness in Rankings and Recommenders [EDBT20], Recommender Systems [SIGMOD'18, EDBT'18, ICDE'15], Personalization [ICDE'10, ICDE'07, VLDB'05].

ACKNOWLEDGMENTS

This work has been partially funded by the European Union's Horizon 2020 research and innovation program (grant agreement No 863410)

REFERENCES

[1] Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A comparative survey of recent natural language interfaces for databases. *VLDB J.* 28, 5 (2019), 793–819.
[2] Ambiguity [n.d.]. Ambiguity. https://stanford.io/2YXcECi.
[3] Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases - an introduction. *Natural Language Engineering* 1, 1 (1995), 29–81. https://doi.org/10.1017/S135132490000005X
[4] DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2020. RYANSQL: Recursively Applying Sketch-based Slot Fillings for Complex Text-to-SQL in Cross-Domain Databases. arXiv:2004.03125 [cs.CL]
[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]
[6] Kedar Dhamdhere, Kevin S. McCurley, Ralfi Nahmias, Mukund Sundararajan, and Qiqi Yan. 2017. *Analyza: Exploring Data with Conversation.* ACM.
[7] Li Dong and Mirella Lapata. 2016. Language to Logical Form with Neural Attention. arXiv:1601.01280 [cs.CL]
[8] Li Dong and Mirella Lapata. 2018. Coarse-to-Fine Decoding for Neural Semantic Parsing. arXiv:1805.04793 [cs.CL]
[9] Orest Gkini, Theofilos Belmpas, Yannis Ioannidis, and Georgia Koutrika. 2021. An In-Depth Benchmarking of Text-to-SQL Systems. In *SIGMOD Conference.* ACM.
[10] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. arXiv:1905.08205 [cs.CL]
[11] Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-SQL: reinforce schema representation with context. arXiv:1908.08113 [cs.CL]
[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (12 1997), 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735
[13] Binyuan Hui, Xiang Shi, Ruiying Geng, Binhua Li, Yongbin Li, Jian Sun, and Xiaodan Zhu. 2021. Improving Text-to-SQL with Schema Dependency Learning. arXiv:2103.04399 [cs.CL]
[14] Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization. arXiv:1902.01069 [cs.CL]
[15] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a Neural Semantic Parser from User Feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, 963–973.
[16] Fei Li and H. V. Jagadish. 2014. Constructing an Interactive Natural Language Interface for Relational Databases. *PVLDB* 8, 1 (Sept. 2014), 73–84.
[17] 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.* Association for Computational Linguistics, Online, 4870–4888. https://doi.org/10.18653/v1/2020.findings-emnlp.438
[18] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-Task Deep Neural Networks for Natural Language Understanding. arXiv:1901.11504 [cs.CL]
[19] Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. 2020. Hybrid Ranking Network for Text-to-SQL. arXiv:2008.04759 [cs.CL]
[20] Jianqiang Ma, Zeyu Yan, Shuai Pang, Yang Zhang, and Jianping Shen. 2020. Mention Extraction and Linking for SQL Query Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Association for Computational Linguistics, Online, 6936–6942. https://doi.org/10.18653/v1/2020.emnlp-main.563
[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL]
[22] Amihai Motro. 1986. Constructing Queries from Tokens. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data* (Washington, D.C., USA) *(SIGMOD '86).* Association for Computing Machinery, New York, NY, USA, 120–131. https://doi.org/10.1145/16894.16866
[23] Notes on Ambiguity [n.d.]. Notes on Ambiguity. http://bit.ly/2YTLFeR.
[24] Debaditya Pal, Harsh Sharma, and Kaustubh Chaudhari. 2020. Data Agnostic RoBERTa-based Natural Language to SQL Query Generation. arXiv:2010.05243 [cs.AI]
[25] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP).* 1532–1543. http://www.aclweb.org/anthology/D14-1162
[26] P. J. Price. 1990. Evaluation of Spoken Language Systems: the ATIS Domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990.* https://www.aclweb.org/anthology/H90-1020
[27] Ohad Rubin and Jonathan Berant. 2020. SmBoP: Semi-autoregressive Bottom-up Semantic Parsing. arXiv:2010.12412 [cs.CL]
[28] Diptikalyan Saha, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R. Mittal, Fatma Özcan, IBM Research. Bangalore, and IBM Research. Almaden. 2016. *ATHENA: An Ontology-Driven System for Natural Language Querying over Relational Data Stores.* VLDB. http://www.vldb.org/pvldb/vol9/p1209-saha.pdf
[29] Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. 2018. IncSQL: Training Incremental Text-to-SQL Parsers with Non-Deterministic Oracles. arXiv:1809.05054 [cs.CL]
[30] Alkis Simitsis, Georgia Koutrika, and Yannis Ioannidis. 2008. Précis: from unstructured keywords as queries to structured databases as answers. *The VLDB Journal* 17, 1 (2008), 117–149.
[31] 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.CL]
[32] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. arXiv:1911.04942 [cs.CL]
[33] 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.CL]
[34] Nathaniel Weir, Prasetya Utama, Alex Galakatos, Andrew Crotty, Amir Ilkhechi, Shekar Ramaswamy, Rohin Bhushan, Nadja Geisler, Benjamin Hättasch, Steffen Eger, Ugur Çetintemel, and Carsten Binnig. 2020. DBPal: A Fully Pluggable NL2SQL Training Pipeline. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 2347–2361.
[35] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv:1609.08144 [cs.CL]
[36] Xiaojun Xu, Chang Liu, and Dawn Song. 2017. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning. arXiv:1711.04436 [cs.CL]
[37] Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. SQLizer: Query Synthesis from Natural Language. *PACMPL*, Article 63 (2017), 26 pages.
[38] Pengcheng Yin and Graham Neubig. 2017. A Syntactic Neural Model for General-Purpose Code Generation. arXiv:1704.01696 [cs.CL]
[39] Pengcheng Yin, Graham Neubig, Wen tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. arXiv:2005.08314 [cs.CL]
[40] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. TypeSQL: Knowledge-based Type-Aware Neural Text-to-SQL Generation. arXiv:1804.09769 [cs.CL]

[41] Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing. arXiv:2009.13845 [cs.CL]

[42] Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018. SyntaxSQLNet: Syntax Tree Networks for Complex and Cross-DomainText-to-SQL Task. arXiv:1810.05237 [cs.CL]

[43] Tao Yu, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter S Lasecki, and Dragomir Radev. 2019. CoSQL: A Conversational Text-to-SQL Challenge Towards Cross-Domain Natural Language Interfaces to Databases. arXiv:1909.05378 [cs.CL]

[44] 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. 2019. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. arXiv:1809.08887 [cs.CL]

[45] Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. SParC: Cross-Domain Semantic Parsing in Context. arXiv:1906.02285 [cs.CL]

[46] Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. arXiv:1709.00103 [cs.CL]