# Translating Natural Language Questions to SQL Queries

Stanford CS224N Custom Project

**Omkar Salpekar**
Department of Computer Science
Stanford University
omkars@stanford.edu

## Abstract

Sequence-to-sequence models have performed well at the Text-to-SQL task on datasets such as WikiSQL. However, most prior work does not examine generalizability of the models to unfamiliar table schemas. We build on the ideas introduced by Chang et al. [1] to improve a sequence-to-sequence dual-task learning model by generalizing better on a zero-shot testbed, which consists of schemas the model has never encountered before. We use the pre-trained BERT-based TAPAS transformer model to encode more expressive table representations for the schema, in addition to the existing BiLSTM-based encodings. Additionally, we use techniques from semantic parsing research such as the coverage mechanism and more flexible attention algorithms to propose a model that achieves a 5+% accuracy improvement over the base dual-task sequence-to-sequence model on the zero-shot test set.

## 1 Key Information

- Mentor: Shikhar Murty
- External Collaborators: N/A
- Sharing Project: N/A

## 2 Introduction

The proliferation of inexpensive hardware and cloud services has enabled vast amounts of data to be stored in databases and data warehouses globally. Extracting insights from this data requires an intuitive understanding of Structured Query Language (SQL), a language which allows the composition of potentially complex and highly nested declarative queries to obtain results from across data sources. Since learning to compose complex SQL queries may restrict many potential users from interacting with data, much work has gone into enabling an alternative and more intuitive interface to query data in lieu of SQL: natural language questions in English.

Translating natural language English questions to valid SQL queries that can be issued to a database can be posed as sequence-to-sequence neural machine translation (NMT) problem with some unique properties. These unique properties stem from the relatively standard template of SQL queries that most Text-to-SQL datasets like WikiSQL [2] follow, as seen in Listing 1.

Listing 1: SQL Structure (* indicates there may be multiple WHERE clauses)

```
SELECT $AGG $SEL_COLUMN FROM table_id WHERE ($COND_COL $COND_OP $COND_VAL)*
```

Given a standardized template, the translation problem can be thought of as a sketch/slot-filling problem where only the aggregation and column names in the SELECT clause and the condition

columns, values, and operations in the WHERE clause must be predicted. The Text-to-SQL task involves translating a natural language English question $Q$ into a SQL query $Y$ given a table schema $C$ by correctly filling the specified slots

While it is clear that accurate Text-to-SQL models could have a massive democratizing effect on data analytics, they are currently not effective enough to be used in a production environment. The primary issue is generalization - most existing models overfit to table schemas encountered during training, and thus cannot generalize well to unfamiliar schemas. Chang et al. is among the first works to address this generalization issue by proposing a dual-task model architecture and introducing a zero-shot testbed, which evaluates models on how well they translate queries from unfamiliar schemas (hence the "zero-shot" nature of the testbed) [1].

In this work, we explore a variety of techniques from semantic parsing, transformers, and attention to propose a model superior to the base dual-task model in Chang et al. Specifically, we use the coverage mechanism to prevent output sequence repetition, a modified loss function that includes coverage vectors for better regularization, Bahdanau attention, and pre-trained BERT-based schema encodings to achieve a **5+%** improvement in query execution accuracy on the zero-shot testbed over the model proposed in Chang et al.

## 3 Related Work

There is a rich body of research that explores translating natural language questions into SQL queries. Early work in this area began with logic-based approaches to the ATIS (Air Traffic Information Service) [3] task and GEO queries [4], although both of these assumed a standard schema since the translation systems were specific to a database. More recent work in the field includes sequence to sequence models combined with Reinforcement Learning to generate SQL queries in Seq2SQL [2]. Seq2SQL consists of an augmented pointer network to learn the aggregations in the SELECT and WHERE clauses as well as the column names, and separately determines the randomly ordered WHERE conditions by using a policy gradient. This work also introduced the WikiSQL dataset which contains pairs of natural language questions and SQL queries for a wide variety of schemas, unlike the standardized schemas in ATIS and GEO.

SQLNet produced better accuracy than Seq2SQL models by removing reinforcement learning and sequence-to-sequence methods and replacing them with a dependency graph approach that took advantage of the standard structure of SQL queries and column attention [5]. Newer work like TypeSQL [6] used the types of SQL query operators (such as column names, aggregators, condition operands, and condition values) to pose a slot-filling problem that outperformed SQLNet, as well as took advantage of the typing information in the table schema to learn expressive table encodings. Techniques like Coarse-to-Fine Decoding [7] [8] introduced multiple layers of decoding for higher- and lower-level understanding to better learn semantic relationships. SyntaxSQLNet combined column-attention encoders with syntax-tree based decoders to support more complex SQL query operators than previous work, such as nested queries and joins [9].

The Zero-Shot Text-to-SQL paper by Chang et al. incorporates many of the techniques described above but focuses much more on generalizability. This paper presents a far more generalizable model for Text-to-SQL translation that will work across a variety of schemas. While other work also explores generalizability in Text-to-SQL, such as that by Suhr et al. [10] on even more challenging datasets such as Spider [11], the results from these works suggest there is significant room for improvement in Text-to-SQL models generalizing to unfamiliar schemas.

## 4 Approach

We propose a model that builds upon the dual-task learning architecture in Chang et al. with new techniques such as the coverage mechanism in attention computations with regularizing coverage loss, BERT/TAPAS-based embeddings as pre-trained table representations, and Bahdanau attention.

### 4.1 Baseline

Seq2SQL kicked off much of the Deep Learning-based work on the Text-to-SQL task [2], and newer work in the field use Seq2SQL as a baseline. Briefly, Seq2SQL consists of an augmented pointer

network to learn the SELECT and WHERE aggregations as well as the column names, and separately determines the randomly ordered WHERE conditions by using a policy gradient.

### 4.2 Primary SQL Generation Task

The dual-task learning architecture in our proposed model consists of a main task for predicting the SELECT clause and the WHERE clause, as well as an auxiliary task that learns the mapping between conditional values and conditional columns in the WHERE clause. We build on top of the code provided by Chang et al. (a link to the original code base is provided in Appendix A.1), making significant changes to enable the approaches described below.

Per Table 5.1, the model input consists of a question $Q$ and a typed table schema $C$. The natural language question $Q$ is encoded with a standard bidirectional LSTM [12] to produce hidden representations $h_t^q$. We implemented a new preprocessing pipeline for encoding the table schema $C$, since we hypothesized learning better table representations should improve generalizability. The question $Q$ and schema $C$ are padded to a standard length, tokenized to add standard separators, and passed as input into the pre-trained TAPAS transformer model [13]. TAPAS is a BERT-based transformer model [14] [15] that is focused on question answering, in that it takes queries and a table (from which it infers a schema given the column index) and outputs answers to those queries. Since we are interested in translating to SQL, we run inference with TAPAS on only the table schema and all queries corresponding to that table in order to extract pre-trained table representations from the transformer's encoder. These pre-trained representations pass through an additional linear layer that learns a relative weighting of different aspects of the table schema (relative importance of columns and their types) to produce table encodings $h^c$. The preprocessing pipeline was implemented from scratch, and we used the TAPAS transformer model in the Huggingface library [16].

We next combine the question encoding $h_T^q$ and table encoding $h^c$ into hidden representations $H^c$ and $H^q$ using bi-attention. $h_1^q$ and $h_T^q$ are concatenated to form the final WHERE clause encoding $q^{WHERE}$. $H^q$ is passed into an attentive pooling layer [17], which utilizes the coverage mechanism [18] [19] (implemented from scratch, explained in the following paragraphs), to form a final SELECT clause encoding $q^{SEL}$. The encoder diagram and preprocessing pipeline are found in Figure 1.

Our model uses 3 different decoders: a classifier decoder for predicting the aggregation in the SELECT clause, a pointer decoder for predicting column names [20], and a coarse-to-fine decoder for predicting the WHERE clause [7]. The decoders use Bahdanau (additive) attention [21] instead of Luong (multiplicative) attention [22] attention to better recall features in the encodings. Additionally, we address the common repetition problem in seq-to-seq models by using the coverage mechanism in the attention computations, which is described in the equations below.

Assume $a^t$ models the attention distribution, computed by taking the softmax of attention scores $e_i^t$. The attention scores are computed using Bahdanau attention as follows:

$$\begin{aligned} e_i^t &= v^T tanh(W_h h_i + W_s s_t + b) \\ a^t &= softmax(e^t) \end{aligned} \tag{1}$$

We then introduce a coverage vector, which is a running sum of the attention distributions of all the previous time steps:

$$\begin{aligned} c^t &= \sum_{t'=0}^{t-1} a^{t'} \\ c^0 &= 0 \end{aligned}$$

We modify the attention computation to use this coverage vector. Specifically, we incorporate the coverage vector into the MLP when computing the Bahdanau attention scores:

$$e_i^t = v^T tanh(W_h h_i + W_s s_t + w_c c_i^t + b) \tag{2}$$

As such, attention scores from previous time steps influence the attention scores at the current time step more, which can maintain context from earlier in the sequence. We also incorporate coverage

into the loss function by adding a coverage loss $covloss$ scaled by some hyperparameter $\beta$ to the main loss function. Below is the coverage loss term for a decoder $d$ and an element $t$ in the input sequence.

$$covloss_t^d = \sum_i min(a_i^t, c_i^t) \tag{3}$$

The total coverage loss will involve summing these losses across sequence length, as well as across the 3 decoders. Assume the set $D$ consists of layers using the coverage mechanism (decoders and attentive pooling). We can then compute the total coverage loss as follows:

$$L_{coverage} = -\sum_{d \in D} \sum_{t=1}^{T} covloss_t^d \tag{4}$$

The final encodings $q^{SEL}$, $q^{WHERE}$, and $H^c$ are passed into the attentive decoders. The classifier decoder uses $q^{SEL}$ to predict AGG in the SELECT clause. The pointer decoder, which can point to a specific token in the input sequence instead of taking a weighted average of them, is useful for predicting column names since these are usually present verbatim in both the input question and schema. As such, the pointer takes both $q^{SEL}$ and $H^c$ as input to predict SELECT column names. Lastly, $q^{WHERE}$ is passed to the classifier and then to the coarse-to-fine decoder to predict conditional phrases in the WHERE clause. The decoder diagram can be found in Figure 2.

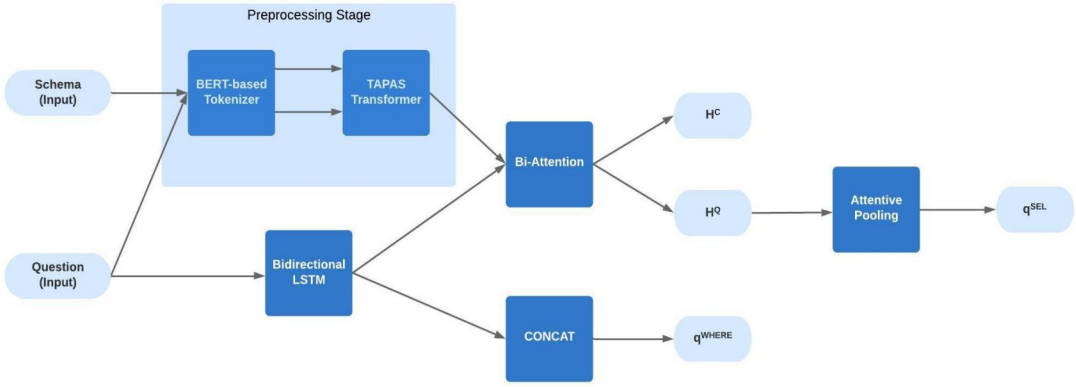


Figure 1: Architecture of the encoder and preprocessing stage of our proposed model. Note that the Question (Q) and Schema (C) are the model inputs, and 3 separate encoded representations ($H^c$, $q^{SEL}$, and $q^{WHERE}$) are encoder outputs.

### 4.3 Auxiliary Mapping Task

The auxiliary task will predict a mapping between condition columns and condition values in the WHERE clause. A Name-Entity Recognition technique called BIO tags [23] is used to label question words as entities, values, or neither - producing a tag vector $y^{tag}$. The tagged words are passed into the pointer decoder along with $H^c$ to predict the pairs (called $y^{map}$) of column names/values of each WHERE clause, which are used with conditional operations predicted by the main task to construct each WHERE condition. While this same technique is used in Chang et al., our $H^c$ table representations are derived from the TAPAS-based preprocessing pipeline and the pointer decoder generalizes better with the coverage mechanism. A diagram of this task can be found in Figure 4 in the Appendix.

The primary task uses cross entropy loss to compare the expected and predicted filled slots $y^{op}$. The cross-entropy loss is also used for the auxiliary task, in which we take the sum of individual cross

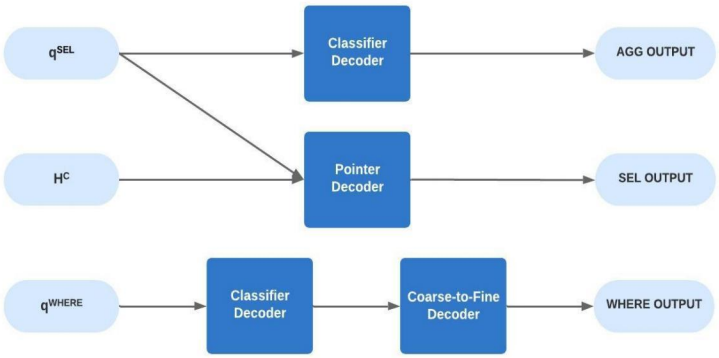


Figure 2: Architecture of the decoder. Note the decoders are using Bahdanau attention and the coverage mechanism as described in Section 4.2.

entropy losses for $y^{tag}$ (to optimize the model for correct NER tagging) and $y^{map}$ (to generate the correct column/value pairs). The total loss function of the model is a weighted sum of the losses of the 2 tasks and the coverage loss, as described in the equations below.

$$L_{primary} = -\sum_{i=1}^{|Y|} y_i^{op} \log \hat{y}_i^{op} \tag{5}$$

$$L_{aux} = -\sum_{i=1}^{|Q|} y_i^{tag} \log \hat{y}_i^{tag} - \sum_{i=1}^{K} y_i^{map} \log \hat{y}_i^{map} \tag{6}$$

$$L_{total} = L_{primary} + \alpha L_{aux} + \beta L_{coverage} \tag{7}$$

# 5 Experiments

## 5.1 Data

This work used the WikiSQL dataset from Salesforce [2], which consists of over 80K examples drawn from nearly 20K unique tables on Wikipedia. Concretely, each example consists of 3 items: a natural language English question $Q$, a typed table schema $C$, and a SQL query $Y$. $Q$ and $C$ are inputs to the model, and $Y$ is the output. One example from the dataset is shown in Table 5.1. Note that the task does not require selecting a table name, so the FROM clause is not present in the examples.

| | |
|---|---|
| Question (Q) | What was the time for Peter Burwick of team Suzuki? |
| Table Schema (C) | Rank (real) \| Rider (str) \| Team (str) \| Speed (real) \| Time (real) |
| SQL Query (Y) | SELECT Time WHERE Rider='Peter Burwick' AND Team='Suzuki' |

Table 1: This table displays an input and output example from Chang et al. Note that the question and table schema are both input and the SQL Query is the output.

The exact data format of the examples is a nested JSON with column names, SELECT aggregations, and conditional operators tokenized into indices. A verbatim example can be found in Listing 2 in the appendix. The dataset separately includes a JSON for the tables, which specifies the schema and

a few rows of data for each table. The actual data for the tables is used to verify that the predicted SQL queries can return the correct results when run on a small database.

In accordance with our unique evaluation criteria, the dataset was then partitioned to create a zero-shot testbed. There were approximately 65K training examples, 10K validation examples, and 5K test examples. The testbed contained queries from over 1500 schemas that were not present in the training and validation sets.

### 5.2 Evaluation method

We compute six qualitative evaluation metrics on the zero-shot testbed. All of these are computed automatically in an evaluation script run after training. Of these, the Query-Match Accuracy ($ACC_{qm}$) and Query-Execution Accuracy ($ACC_{ex}$) are the most important and holistic measures of accuracy on the Text-to-SQL task. Their details are described below:

- **Query-Match Accuracy:** The percentage of matches between the predicted and ground truth queries - checking aggregators, column names, condition values, etc. If there are multiple WHERE conditions, their order does not matter.
- **Query-Execution Accuracy:** The percentage of matches between the tables returned when the predicted and ground truth query are executed on the given data. This is a more forgiving metric than Query-Match accuracy since multiple queries can return correct results.

The remaining four metrics evaluate accuracy on specific slots or clauses in the predicted SQL query. These are generally less important than the above two metrics, but large variations in these metrics may suggest certain experimental models are significantly better or worse are predicting specific slots. This additional insight can help suggest which parts of the model can be further improved in the debugging and experimentation process.

- **Aggregator Accuracy:** Compares the SELECT aggregators (COUNT, SUM, etc.) between predicted and ground truth queries
- **SELECT Accuracy:** Compares the SELECT clause (column names and SELECT aggregators) between predicted and ground truth queries
- **WHERE Accuracy:** Compares the WHERE components (condition columns, operations, and values) between the predicted and ground truth queries.
- **COLUMN Accuracy:** Compares all column names (in both SELECT and WHERE) between the predicted and ground truth queries.

### 5.3 Experimental details

The experiments build off the code from the base dual-task model in Chang et al. [1] with new functionality implemented from scratch and integrated into the model per experiment, such as various attention types, the coverage mechanism, and the TAPAS preprocessing pipeline. The model is implemented in PyTorch [24].

The experiments were run for 45 epochs with a 0.001 learning rate that starts decaying by 2% after the 8th epoch. While we did not extensively tune hyperparameters, we used $\alpha = \beta = 0.01$ in our weighted cross-entropy based loss function described in Equation 7 in Section 4.2. We use the Adam optimizer [25] for gradient descent. Model training took approximately 6-8 hours on a single GPU.

The existing model and TAPAS could not both fit in GPU memory (12.0GB on a single GPU), and running either on CPU was prohibitively slow, so the table representations were pre-generated using TAPAS inference on GPU and stored in a JSON. This JSON was later parsed into a large PyTorch Float Tensor, used to populate an embedding layer, and serialized, so that it could be used for batched embedding lookups during training. The embedding layer itself is frozen during training.

### 5.4 Results

Results from select experiments are shown in Table 2. Seq2SQL is the baseline, and (DT) Base indicates the model from Chang et al. as is. The rest of the experiments denote changes made on top of the base dual-task model.

| Model | $ACC_{qm}$ | $ACC_{ex}$ | $ACC_{agg}$ | $ACC_{sel}$ | $ACC_{where}$ | $ACC_{col}$ |
|---|---|---|---|---|---|---|
| Seq2SQL | - | 59.4 | 90.1 | 88.9 | 60.2 | 69.12 |
| (DT) Base | 62.06 | 70.70 | 90.29 | 89.38 | 74.26 | 79.06 |
| (DT) Bahdanau | 62.83 | 70.72 | 90.43 | 89.15 | 75.30 | 79.83 |
| (DT) Typed Attn + Cov | 64.11 | 71.84 | 90.25 | 90.00 | 77.18 | 82.42 |
| (DT) Cov | 64.73 | 72.35 | 90.06 | 90.16 | 77.57 | 82.50 |
| (DT) Cov + Cov Loss | 65.09 | 72.79 | 90.08 | 89.89 | 77.41 | 82.40 |
| (DT) TAPAS + Cov + Cov Loss | **69.61** | **76.54** | 90.18 | **91.58** | **81.12** | **84.41** |

Table 2: Accuracy results from select experiments. Numbers are all percentages. DT stands for dual-task, referring to the dual-task seq2seq model proposed in Chang et al. The last 5 experiments all use Bahdanau attention. "Cov" indicates the Coverage Mechanism used in the Attention Computations.

The model using Coverage and Coverage Loss in the decoders and Attentive Pooling layer along with TAPAS-generated table representations showed the best accuracy across the board, and particularly for the 2 critical metrics: $ACC_{qm}$ and $ACC_{ex}$. While it was expected that generalizability-focsed techniques would yield accuracy benefits, the 5.84% improvement over the base model from Chang et al. is an encouraging result. This makes sense as the coverage mechanism provides a powerful tool for preventing incorrect repetition in the output sequence. Additionally, the TAPAS-based table encodings take advantage of a powerful, pre-trained BERT-based transformer as opposed to a simple bidirectional LSTM that is learned from scratch in the Chang et al. model. The model that only used Coverage also displayed significant improvements over the base Dual-Task model from Chang et al., and adding Coverage Loss to this further increased performance. The type-aware attention, inspired by TypeSQL [6], uses a weighted average of the individual word vectors in the schema to form a final table representation, yet this experiment combined with coverage yielded worse performance than the experiment with coverage alone.

## 6 Analysis

In order to more accurately judge the generalizability of the model, we conduct a similarity study on the zero shot testbed. Specifically, we sought to determine how different test set table schemas really are from those in the training set. We then use a measure of this difference to determine whether the familiarity of schemas is correlated with accuracy on the test set.

In order to do this, we start with the BERT-based table encodings generated in the pre-processing pipeline using the TAPAS transformer. Next, for each example the in the test set, we computed the average distance between its table encoding and the table encodings of its three nearest neighbors in the training set. We can call this average distance the "uniqueness score" of each table schema, since lower scores indicate smaller distances from familiar schemas, meaning the test schema is more familiar and less unique. The nearest neighbor computation was done using Facebook's FAISS library [26], in which we created a searchable binary index of all the training set table encodings and computed both cosine similarity and L2 norm as distance metrics.

We then computed the query-match accuracy of our proposed model on the testbed grouped by table schema. Combining the accuracy observed per table schema with the uniqueness score of each schema results in the plot in Figure 3.

From the plot, we can infer that truly unique schemas, while not altogether too common in the test set, do typically result in lower accuracy. This makes intuitive sense since it is difficult for most

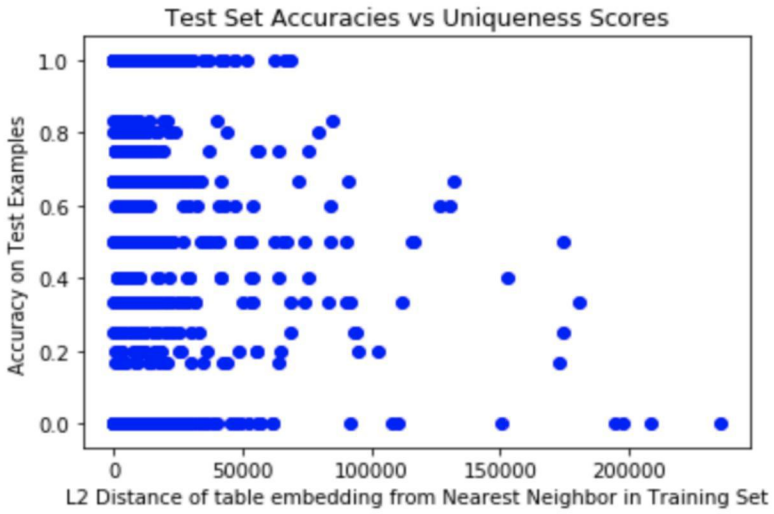


Figure 3: Similarity Study plot. Each point represents a table schema in the test set. The Y axis depicts the accuracy for each schema in the test set, and the X axis depicts the distance from the schemas' nearest neighbor in the training set. Note the clear trend that very unique schemas, while relatively rare in the test set, see low accuracy.

models to predict accurately when the input example is very different from any example encountered during training, especially given this study is conducted on the most stringent metric ($ACC_{qm}$) that expects a verbatim match between the predicted and gold SQL queries. However, it also suggests that additional work is needed to predict accurately for "long-tail" schemas.

While the model proposed in this work achieved an accuracy improvement over the model in Chang et al., the plot shows there is still room for improvement on the zero-shot testbed for relatively familiar schemas. Many tables with lower uniqueness scores also see low accuracy, suggesting factors apart from table schema are still sub-optimal. Further optimizations outside of generalizability, such as in model architecture and hyperparameter tuning, can contribute to more accurate translations on relatively familiar examples. This may also suggest that improving the question encodings, exploring end-to-end transformer-based methods, and using other techniques that improve SQL translation quality may yield more accuracy benefits on this testbed than trying to improve generalizability to accurately translate with very unique schemas.

## 7 Conclusion

We built an improved Text-to-SQL model that translates natural language English questions to valid SQL queries, while showing better accuracy on a zero-shot testbed than previous work. Our dual-task model uses a pre-trained BERT-based TAPAS transformer model to generate expressive table representations in the encoder. We draw from semantic parsing research by feeding these table encodings, along with question encodings, into a set of generalizable decoders that use Bahdanau attention, the coverage mechanism for better recall and preventing repetition in the translation, and coverage loss. These techniques allow us to demonstrate a 5+% improvement over the base dual-task model architecture in Chang et al. on the zero-shot testbed.

Our similarity study showed that accuracy on examples decreases as the uniqueness of the table schema increases, past a certain threshold. Very unique schemas are still difficult to generalize to, and future work may explore further generalization techniques including using only transformer encodings for both the question and schema or even variations of transformers for the entire task. To improve accuracy on less unique schemas, future work can similarly draw on advancements in transformer networks to produce high-quality translations.

## References


[1] Shuaichen Chang, Pengfei Liu, Yun Tang, Jing Huang, Xiaodong He, and Bowen Zhou. Zero-shot text-to-sql learning with auxiliary task. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7488–7495, Apr. 2020.

[2] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017.

[3] Deborah A. Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

[4] John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, page 1050–1055. AAAI Press, 1996.

[5] Xiaojun Xu, Chang Liu, and Dawn Song. Sqlnet: Generating structured queries from natural language without reinforcement learning. *CoRR*, abs/1711.04436, 2017.

[6] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 588–594, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[7] Li Dong and Mirella Lapata. Coarse-to-fine decoding for neural semantic parsing. *CoRR*, abs/1805.04793, 2018.

[8] Mazen Mel, Umberto Michieli, and Pietro Zanuttigh. Incremental and Multi-Task Learning Strategies for Coarse-to-Fine Semantic Segmentation. *Technologies, special issue on Computer Vision and Image Processing Technologies*, 8(1), 2020.

[9] Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir R. Radev. Syntaxsqlnet: Syntax tree networks for complex and cross-domaintext-to-sql task. *CoRR*, abs/1810.05237, 2018.

[10] Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. Exploring unexplored generalization challenges for cross-database semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8372–8388, Online, July 2020. Association for Computational Linguistics.

[11] 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. 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*, pages 3911–3921, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[13] Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Seattle, Washington, United States, 2020.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

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

[17] C. D. Santos, M. Tan, Bing Xiang, and Bowen Zhou. Attentive pooling networks. *ArXiv*, abs/1602.03609, 2016.

[18] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Coverage-based neural machine translation. *CoRR*, abs/1601.04811, 2016.

[19] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368, 2017.

[20] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks, 2017.

[21] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015.

[22] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015.

[23] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[24] Adam Paszke, S. Gross, Francisco Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, B. Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[26] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

# A Appendix

## A.1 Zero-Shot Base Code

The code provided by Chang et al. can be found here: https://github.com/JD-AI-Research-Silicon-Valley/auxiliary-task-for-text-to-sql. We also also use the Huggingface Transformer library for generating the TAPAS encodings [16].

### A.2 Auxiliary Task Diagram

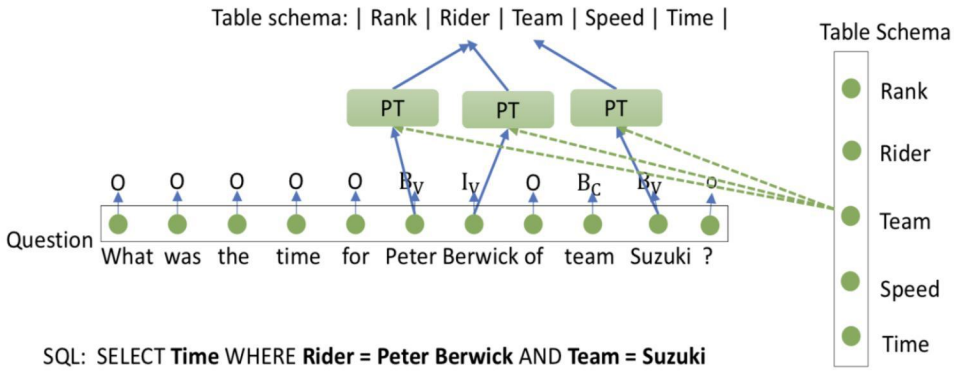


Figure 4: Architecture of the Auxiliary Task Model. This diagram is from Zero-Task paper by Chang et al. An NER technique is used to detect entities in the input question. Each entity is passed into the pointer decoder and attended with the table schema to "point" to corresponding column in the schema, producing a mapping from entities (condition values) to condition columns.

### A.3 WikiSQL Data Format

Listing 2: WikiSQL data format. Note that columns and operations have all been tokenized.

```
{
 "phase": 2,
 "table_id": "2-15496934-1",
 "question": "What is the Serial number of the Locomotive that Entered
     Service in November 1984 and has an Owner of Chicago Freight Car
     Leasing Australia?",
 "sql": {
        "sel": 1,
        "conds": [[2, 0, "november 1984"],
                  [3, 0, "chicago freight car leasing australia"]],
        "agg": 0
       }
}
```