Texila International Journal of Academic Research
ISSN: 2520-3088
DOI: 10.21522/TIJAR.2014.11.01.Art015

Topological Relation Aware Transformer

Nathan Manzambi Ndongala
Department of Computer Science, Texila American University, Guyana

Abstract

We present a Topological Relation Aware Transformer (T-RAT), a specialized head transformer to open sets, an element of the topology τ *generated by the set S, the set of all pre-existing relations between input tokens of the model. From this topological space* (S, τ)*, we present the way to spread each open set to one head of our Transformer. T-RAT improves exact match accuracy in Text-To-SQL challenge (62.09%) without any enhancement of large language models compared to the baseline models RAT-SQL (57.2%) and Light RAT-SQL (60.25%).*

Keywords: *Deep learning, Natural Language Processing, Neural Semantic Parsing, Relation Aware Transformer, RAT-SQL, Text-To-SQL Transformer.*

Introduction

In Text-To-SQL [1-9], The Light RAT-SQL [8] shows the way to reduce the number of preexisting relations in the RAT-SQL framework [7] by preserving the exact match accuracy without any enhancement of pre-trained (LLMs) large language models [10-15]. The limitation of this method is that it can not be suitable for the scenario where we have a lot of pre-existing relations. Since each head in multi-head self-attention must be specialized to at least one relation, we will have a model with high dimensions.

Another con is that Light-RAT-SQL suffers from *over-alignment* as we can see in Figure 1, there is more unnecessary attention on column **"*",** the same with RAT-SQL the column horsepower getting so much attention.

To fix this, we present a new model, the Topological Relation Aware Transformer(T-RAT) inspired by the Topology theory and Relation Aware Transformer [1, 7] where we improve the "spreading algorithm" presented in Light RAT-SQL.

The Topological Relation-Aware Transformer (T-RAT) is a specialized head transformer designed to address the problem of semantic parsing of natural language to SQL queries, particularly in the context of database query tasks. In this paper, we present a novel approach to leveraging the inherent relations between tokens, such as those between natural language questions and database schema elements, to improve the accuracy of text-to-SQL query generation models. T-RAT is built on the foundation of topology theory and Relation Aware Transformer (RAT), which improves the capacity of multi-head self-attention models to capture diverse and contextually relevant relationship patterns within complex data. Our proposed method spreads each open set to one head of the Transformer, enabling the model to capture nuanced relationships between input tokens and improve the accuracy of text-to-SQL query generation.

We believe that RAT-SQL overfits because of many preexisting relations, and Light RAT-SQL misaligned some tokens due to few relations and mixing forward and backward edges.

Received : 08.12.2023 ***Accepted : 27.01.2024*** ***Published on : 31.01.2024***
Corresponding Author : nathanmanzambi@yahoo.fr

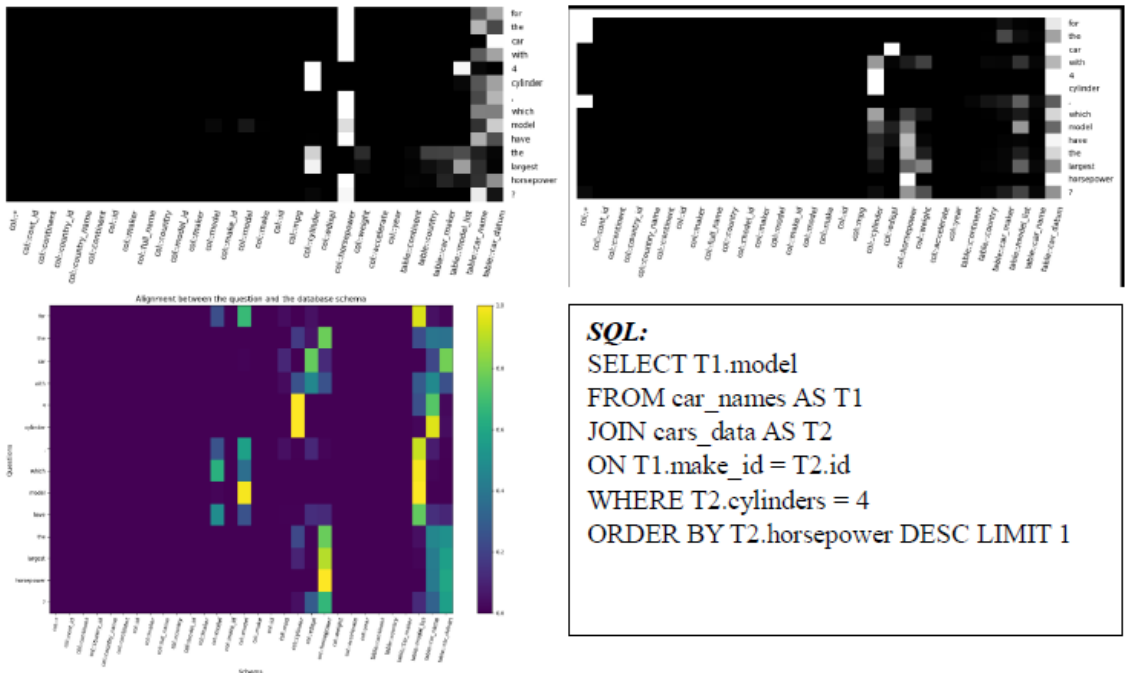

Figure 1. Alignment between the Question "For the Cars with 4 Cylinders, which Model has the Largest Horsepower" and the Database Car_1 Schema (Columns and Tables)

Up-Left: The RAT-SQL – Up-Right: Light RAT-SQL and bottom-left: T-RAT

So, this question remains open: How can the inherent relations between tokens, such as those between natural language questions and database schema elements, be effectively leveraged to improve the accuracy of text-to-SQL query generation models?

The inherent relations between tokens, such as those between natural language questions and database schema elements, can be effectively leveraged to address the problem of semantic parsing of natural language to SQL queries. We hypothesize that by spreading each open set to one head of the Transformer, the T-RAT model can capture nuanced relationships between input tokens, leading to improved accuracy in text-to-SQL query generation models. Additionally, we believe that the topology structure of the pre-existing relations enhances the capacity of multi-head self-attention models to capture diverse and contextually relevant relationship patterns within complex data. This will help to have feature diversification and learning complementary information, ultimately improving the overall accuracy of text-to-SQL query generation models. This hypothesis forms the basis for our exploration of the T-RAT model and its potential to effectively address the challenges of semantic parsing in the context of natural language to SQL queries.

Our contribution and key findings:

1. We present a way to leverage preexisting relations between input tokens in Text-To-SQL to improve the exact match accuracy of RAT-SQL, Light RAT-SQL (without any enhancement of the pre-trained large language models).
2. The topology structure of the pre-existing relations enhances the capacity of multi-head self-attention models to capture diverse and contextually relevant relationship patterns within complex data. This will help to have feature diversification and learning complementary information. So, improving the overall accuracy.
3. By tailoring each head to different elements of the topology, it facilitates a more detailed and nuanced understanding of relationships in the input data, making it a valuable tool in NLP tasks.

A detailed description of the T-RAT model has been presented so far, including the algorithms used to encode and process input data and demonstrate its effectiveness in addressing the problem of semantic parsing of natural language to SQL queries. Our experimental results show that T-RAT achieves higher exact match accuracy compared to baseline models RAT-SQL and Light RAT-SQL.

Overall, T-RAT presents a promising approach to addressing the problem of semantic parsing of natural language to SQL queries in low-resource settings, and our proposed method of spreading open sets through heads provides a valuable tool for capturing nuanced relationships between input tokens.

Related Work

Text-To-SQL

The exploration of Text-to-SQL conversion employing deep learning techniques has witnessed considerable advancements, driven by the expressive power and adaptability of neural network architectures. This section provides an overview of the related work in the domain, highlighting key developments and methodologies.

Sequence-to-Sequence Models

Early forays into leveraging deep learning for Text-to-SQL predominantly featured sequence-to-sequence models [2, 16]. These models, inspired by their success in machine translation, were adapted to map natural language utterances to SQL queries. The encoder-decoder architecture facilitated the capturing of complex linguistic structures, yet challenges persisted in handling the semantic intricacies inherent in SQL generation.

Attention Mechanisms

The integration of attention mechanisms [17-21] marked a significant enhancement in capturing dependencies within the input sequence. Attention mechanisms allowed models to focus on specific parts of the input when generating corresponding SQL tokens, improving the overall contextual understanding. Noteworthy works, such as SQLNet [4, 6, 16], introduced attention mechanisms tailored for the Text-to-SQL task.

Pre-trained Language Models

The rise of pre-trained language models [10-12], [22-24], such as BERT [14, 25-28] and GPT (Generative Pre-trained Transformer) [13, 15], has had a transformative impact on various natural language processing tasks, including Text-to-SQL. Fine-tuning these models for SQL generation tasks demonstrated substantial gains in capturing nuanced linguistic patterns and semantic relationships. GP [29], GAP [30], GRAPPA [31], STRUG [32] are among Pretraining Text-To-SQL.

Semantic Parsing Techniques

Advancements in semantic parsing [33-36] have played a crucial role in refining Text-to-SQL models. Techniques incorporating semantic role labeling and syntax-aware parsing have been explored to imbue models with a deeper understanding of the underlying semantics, enabling more accurate SQL query generation.

Graph-Based Semantic Parsing

Graph-based methods [1, 37-43] have gained prominence for their ability to represent complex relationships and dependencies within a sentence. Dependency graphs or semantic graphs are constructed to capture the syntactic and semantic connections between words. Nodes in the graph correspond to words, and edges encode grammatical or semantic relationships. Graph-based methods facilitate a holistic understanding of sentence structure, enabling effective extraction of formal semantic representations for subsequent processing.

Cross-Domain Adaptability and Multilingual Text-to-SQL

Recent efforts have been directed towards enhancing the adaptability of Text-to-SQL

models across diverse domains and linguistic variations. Enhanced by GPT-4 [44], these models [45, 46] perform high accuracy and are now SOTA in most benchmark.

The availability of high-quality datasets is crucial for training, evaluating, and benchmarking models.

Various datasets have been created to address the complexity of mapping natural language queries to SQL queries accurately: ATIS, Spider, WikiSQL, CoSQL, etc. [47-61].

In summary, the evolution of Text-to-SQL with deep learning has witnessed a shift from traditional sequence-to-sequence models to sophisticated architectures incorporating attention mechanisms, pre-trained language models, and hybrid learning strategies. Ongoing research continues to explore innovative approaches to improve model accuracy, generalization, and adaptability across diverse linguistic and database contexts.

Topology on a Set

Formally, let S be a set and let τ be a family of subsets of S. Then τ is called a topology on S if:

1. Both the empty set and S are elements of τ.
2. Any union of elements of τ is an element of τ.
3. Any intersection of finitely many elements of τ is an element of τ.

If τ is a topology on S, then the pair (S, τ) is called a topological space.

The members of τ are called open sets in S.

If the complement of a subset of **S** is in τ, then the subset is considered closed; that is, its complement is open. A subset of **S** can be neither, both (a clopen set), open, or closed. There is always a closed and an open set, the empty set and **S** itself. A neighborhood of **S** is an open subset of S that includes the point **s** [62].

RAT-SQL

The text-to-SQL encoder should be able to encode database relations in such a way that the semantic parser and model alignment between the database schema and the given query are optimized.

How the inputs (schema: tables & columns, and question) are simultaneously encoded is critical for generalization purposes for unknown database structures.

The RAT-SQL framework [7], built on the relation-aware self-attention mechanism [20] and pointer networks [18], solves the Text-to-SQL challenge's generalization problem.

The relative position distance in input tokens is used as the graph's edge in relation-aware self-attention [18], whereas RAT-SQL uses an embedding of abstract type of existing relations between tokens.

However, both bias the self-attention equation in the same way to inject the graph's edges into their model.

Equations of relation-aware self-attention:

$$e_{i,j}^{h} = \left(\frac{(XW^{q}(XW^{k}+R_{k})^{T})}{\sqrt{d_{k}}}\right) \quad (1)$$

$$\alpha_{i,j}^{h} = softmax\left(e_{i,j}^{h}\right) \quad (2)$$

$$z_{i}^{h} = \sum_{v} \alpha_{i,j}^{h} \left(XW^{v} + R_{v}\right) \quad (3)$$

where matrices W^{q}, W^{k}, *and* W^{v} are trainable parameters in self-attention.

R_{k} and R_{v} are embedding learned from the abstract relations of the input graph in the key space and value space of the transformer [17].

Methods

The Topological Relation Aware Transformer (T-RAT) is a specialized head transformer designed to address the problem of semantic parsing of natural language to SQL queries, particularly in the context of database query tasks. T-RAT achieves this by leveraging the topology structure of pre-existing relations between input tokens and spreading each open set to one head of the Transformer.

T-RAT uses a direct graph to represent the pre-existing relations between input tokens, such as syntax dependency in question tokens, schema linking between the question tokens and column/table tokens, and schema encoding in column and table tokens.

The direct graph consists of different edges that describe the relationships between the tokens, such as forward and backward syntax dependency, value-based linking, name-based linking, foreign key linking, and primary key linking.

T-RAT then uses multi-head self-attention as in the baseline model (Light RAT-SQL) to tailor each head to different elements of the topology, facilitating a more detailed and nuanced understanding of relationships in the input data.

This allows T-RAT to capture diverse and contextually relevant relationship patterns within complex data, leading to improved accuracy.

Problem Definition (Text-To-SQL)

Given a natural language question $\mathbf{Q=q1\ldots q_{|Q|}}$, a database schema $\mathbf{S = <C, T>}$ with columns $\mathbf{C=\{c1,\ldots, c_{|C|}\}}$ and tables $\mathbf{T=\{ t1,\ldots, t_{|T|}\}}$

The objective of text-to-SQL is to predict the SQL query $\boldsymbol{y}$ from the input $<\mathbf{Q, S}>$

The most used model in such a challenge is an encoder and decoder architecture pattern with attention mechanisms between the encoder and decoder.

The encoder encodes input as graph $\mathbf{G=<V, R>}$ where $\mathbf{V=Q\ U\ T\ U\ C}$ are nodes of types { **Q, T, C** } The initial embedding matrix X $\mathbf{R^{Vxd}}$, [**V = Q+T+C,** and **d** is the dimension model] is flattened and the edge $\boldsymbol{R}$ is the known relation between two input tokens.

In this work, our proposed method improves the encoder side, especially, the existing RAT-SQL model [8, 7]. Please refer to [33, 34, 62] works for a thorough description of the decoder side.

Relations in T-Rat: Open Heads

Given a set S = {Question, Linking, Schema} Where each element of S is a set of indexes and each index represents an edge type in the direct graph G.

1. Question= {1, 2}
2. Linking = {3, 4, 5, 6}
3. Schema= {7, 8, 9,., 16}

Table 1. gives more details about each index, the name of the abstract edges, the set where the index belongs, and the description of the edge.

The topology generated by a set **S** of sets can be thought of as the set of all possible unions of elements from the sets in **S**, including the empty set.

The topology generated by **S** would consist of all possible unions of elements from **Question, Linking, Schema,** as well as the empty set.

So, the topology generated by **S** would include:

1. The empty set: ∅
2. The sets Question, Linking, and Schema individually.
3. The unions of pairs of sets, such as Question ∪ Schema, Question ∪ Linking, and Schema ∪ Linking.
4. The union of all three sets: Question ∪ Linking ∪ Schema.

This topology represents all the possible combinations and subsets that can be formed by taking elements from the sets **Question, Linking, and Schema**, including the individual sets and the empty set.

This generated topology has 8 elements, and each head of T-RAT will be specialized to one open set. For example: suppose that the second head will be specialized to the open set "**Question**" That means this head will learn only: *SD-forward* and *SD-backward* because these are 2 edges in this open set as described in Table 1.

Algorithms

Figure 2. Shows different modules and components involved in T-RAT.

Algorithm 1: Building Open Sets

This algorithm initializes a list to represent different sets such as "Question," "Schema," and "Linking." It then creates a list of indices to map the indices of edges corresponding to each set, effectively building open sets for the input data.

Algorithm

1. Initialize a list **S** to represent different sets such as "Question," "Schema," and "Linking."
2. Create a list of R_indices to map the indices of edges corresponding to each set:
   a. For "Question," the indices are [1, 2].
   b. For "Schema," the indices are [7, 8, 9, 10, 11, 12, 13, 14, 15, 16].
   c. For "Linking," the indices are [3, 4, 5, 6].
3. Define a list **set_combinations** to store all possible combinations of these set indices, creating a topology (excluding the empty set):
   a. Generate combinations from R_indices of size 1, 2, and 3 to encompass all possible combinations.

The time complexity of Algorithm 1 is $O(n)$, where n is the number of elements in the sets "Question," "Schema," and "Linking." The algorithm iterates through the elements of the sets to initialize the list R and create a list of R_indices, which involves simple iteration and mapping operations.

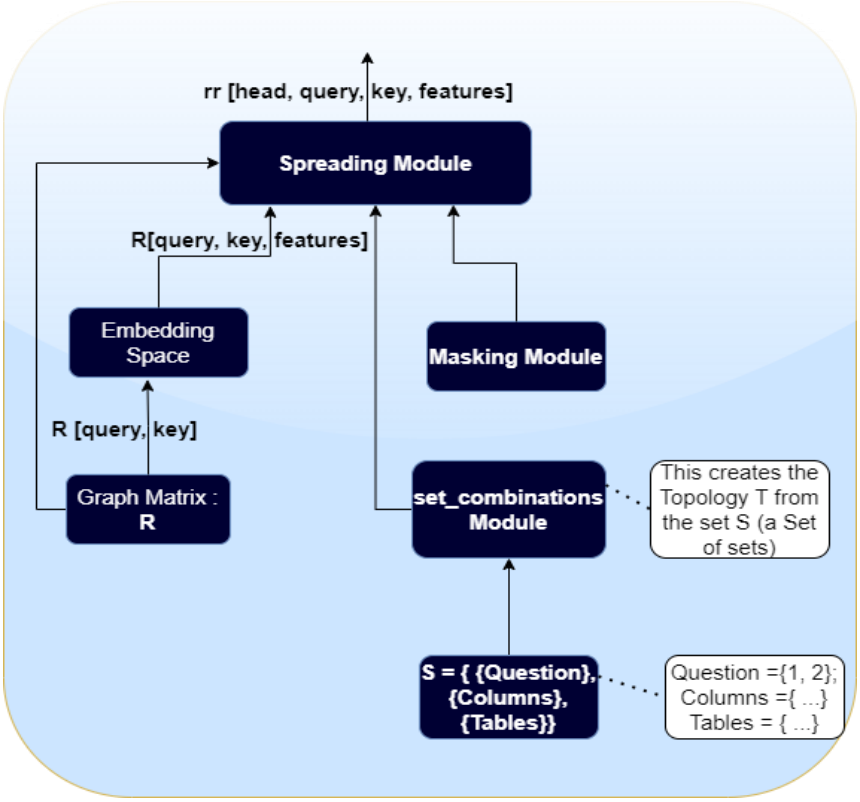

Figure 2. Different Modules of Spreading Open Sets through Heads

The space complexity of Algorithm 1 is O(n), as it requires space to store the sets "Question," "Schema," and "Linking," as well as the lists R and R_indices.

Algorithm 2: Masking Function

The masking function takes three parameters: relation, space_embedding, and set_indices. It initializes a mask tensor with the same shape as the relation and sets it to all zeros. It then iterates through each index in set_indices, checks if the relation is equal to space_embedding[i], and accumulates these matches into the mask.

Algorithm

1. Define a masking function that takes three parameters: relation, space_embedding, and set_indices.
2. Initialize a mask tensor with the same shape as the relation and set it to all zeros.
3. Iterate through each index i in set_indices:

a. Check if the relation is equal to space_embedding[i] and accumulate these matches into the mask.

4. Return the resulting mask.

The time complexity of Algorithm 2 is O(n), where n is the number of elements in the set_indices list. The algorithm iterates through the set_indices list to check if the relation is equal to space_embedding[i] and accumulates these matches into the mask. This involves simple iteration and comparison operations.

The space complexity of Algorithm 2 is *O*(m), where m is the size of the relation tensor. The algorithm requires space to store the mask tensor, which has the same shape as the relation tensor.

Table 1. Description of Different Edges in Direct Graph used in T-RAT

Index	**Edge label**	**SET**	**Description**	**X**	**Y**
1	SD-forward	Question	**Y** is the source word of **X** under syntax dependency	Q	Q
2	SD-backward		**Y** is the target word of **X** under syntax dependency	Q	Q
3	VBL forward	Linking	Value-based-Linking forward: question **X** references any values found in the database and so participates in the SQL query	Q	C
4	VBL backward		Value-based-Linking backward: question **Y** references any values found in the database and so participates in the SQL query	C	Q
5	NBL forward		Name-based linking forward refers to exact or partial occurrences of either the table name or Column name **Y** in question **X**	Q	T or C
6	NBL backward		Name-based linking backward refers to exact or partial occurrences of either the table name or Column name **X** in question **Y**	T or C	Q
7	FK-Forward	Schema	The table **X** has a column **Y** as a foreign key	T	C
8	FK-Backward		The column **X** is a foreign key to the table **Y**	C	T
9	DF-Forward		There is a functional dependency between Column **X** and Column **Y**	C	C
10	DF-Backward		There is a functional dependency between Column **Y** and Column **X**	C	C
11	Belong-To-Forward		Column **Y** belongs to Table **X**	T	C
12	Belong-To-Backward		Column **X** belongs to Table **Y**	C	T
13	PK-Forward		Column **X** is the primary key of Table **Y**	C	T
14	PK backward		Column **Y** is the primary key of Table **X**	T	C
15	TT-forward		Table **X** and Table **Y** are linked by a foreign key	T	T
16	TT-backward		Table **Y** and Table **X** are linked by a foreign key	T	T

The index 0 is linked to “None”, the edge label describing that there is no linking between two tokens. Q: Question | T: Table | C: Column

Algorithm 3: Spreading Open Sets Through Heads

This algorithm operates by initializing dimensions based on the shape of relations and the number of heads in the model. It then creates an empty tensor and, for each head, creates a mask using the masking function with specific set indices. It multiplies the relation tensor by the mask and stores the result, effectively spreading open sets through heads to specialize the relations in the model.

Algorithm

Given:

1. A tensor of relations: relations.
2. Embeddings Space: space_embedding.
3. Algorithm1. Combinations representing the topology: set_combinations.
4. Algorithm 2. Masking module.

The algorithm operates as follows:

1. Initialize dimensions: q, k, f based on the shape of relations.
2. Initialize "head" based on the number of heads in the model.
3. Create an empty tensor rr with dimensions (head, q, k, f).
4. For each head i:
   a. If i=0:
      i. Create a mask using the masking function with only the index 0.
      ii. Multiply the relation tensor relations by the mask and store it in rr [i].
   b. Else:
      i. Retrieve the set of indices set_indice from set combinations[i−1].
      ii. Create a mask using the masking function with set_indices.
      iii. Multiply the relation tensor relations by the mask and store it in rr[i].
5. Return the tensor rr representing the relations with specialized heads.

In Algorithm 3, the Embeddings Space denoted as "space_embedding," refers to the embeddings of the abstract type of existing relations between tokens. These embeddings are learned during the training process and are used to represent the abstract relationships between the input tokens in the direct graph used by T-RAT. The space_embedding is used in the masking function to check if the relation between two tokens matches a specific type of relationship represented by an index in the set_indices list. If the relation matches the type of relationship represented by an index, the corresponding element in the mask tensor is set to 1, indicating that the relation should be included in the output tensor rr.

The time complexity of Algorithm 3 is $O(h * n^2)$, where h is the number of heads in the model and n is the number of elements in the set_combinations list. The algorithm iterates through each head and "sets combination" to create a mask using the masking function and multiplying the relation tensor by the mask. This involves nested iteration and multiplication operations.

The space complexity of Algorithm 3 is $O(h * m^2)$, where m is the size of the relation tensor. The algorithm requires space to store the rr tensor, which has dimensions (head, q, k, f), and the mask tensor, which has the same shape as the relation tensor.

These algorithms are integral parts of the proposed Topological Relation-Aware Transformer (T-RAT) model and are designed to handle the encoding and processing of input data, particularly in the context of text-to-SQL tasks and relation-aware self-attention mechanisms.

Experiments

Dataset

The dataset used is Spider [50]. Spider is a comprehensive text-to-SQL dataset annotated by 11 Yale students, aiming to facilitate the development of natural language interfaces for cross-domain databases. The dataset comprises 10,181 questions and 5,693 unique complex SQL queries, spanning 200 databases with multiple tables across 138 domains. Spider presents a challenge by featuring diverse SQL

queries and databases in both training and testing sets, requiring systems to excel in generalization to new queries and database schemas. More details about Spider dataset: https://yale-lily.github.io/spider.

Implementation

The same architecture as in the RAT-SQL and Light RAT-SQL with the Spider dataset where we applied Algorithm 3 before computing the relation aware self-attention to make pre-existing relations specialized through heads.

The tokens (questions, column names, and table names) are tokenized and lemmatized using the Stanford NLP toolkit [63].

We use pre-trained word embeddings Glove [64] and then each part (question, columns, tables) is processed with bidirectional LSTMs [65] with 128 hidden sizes with a dropout with a rate of 0:2. On the top of this layer, we have 8 T-RAT layers.

The same configuration as RAT-SQL is used for T-RAT, we set **dx = dz = 256, H = 8**, and employ dropout with a rate of 0:1.

The inner layer dimension of the position-wise feed-forward network is 1024. We use rule embeddings of size 128, node type embeddings of size 64, and a hidden size of 512 inside the LSTM with a dropout of 0:21 inside the decoder.

Hyperparameter

The code was implemented in Pytorch[66] The training is done with Adam [67] as an optimizer with default hyperparameters. We use a batch size of 20 and train for up to 40,000 steps.

Results

The experimental results from Table 2 demonstrate compelling evidence of the enhanced performance of T-RAT models in comparison to the baseline Light RAT-SQL and RAT-SQL model, particularly on the test set of the Spider dataset.

Specifically (in a setting without any enhancement of Pre-trained LLMs), T-RAT achieves an accuracy of 62.09%, while Light RAT-SQL achieves an accuracy of 60.25%, surpassing the accuracy of RAT-SQL, which stands at 57.2%.

These findings underscore the efficacy of the proposed enhancements in T-RAT for improving the accuracy of text-to-SQL query generation models.

By outperforming the baseline RAT-SQL model on the challenging test set of the Spider dataset, T-RAT exhibits promising potential in addressing the complexities of semantic parsing in natural language processing tasks.

We ran 5 random experimentations of T-RAT-SQL, Light RAT-SQL, and RAT-SQL without any pre-trained Large Language Models for comparison purposes (cf. Figure 3).

Table 3 shows the confidence interval of each model, and we fail to reject H_0 for RAT-SQL and Light RAT-SQL but there is strong evidence against H_0 ($p<0.001$) for RAT-SQL and T-RAT.

Table 2. Accuracy of the Spider Dataset

Model	**Dev**	**Test**
(Without any enhancement of a Pretrained Large Language Model: BERT, ELECTRA, etc.)		
IRNet [68]	53.2	46.7
Global-GNN [39]	52.7	47.4
IRNet V2 [68]	55.4	48.5
RAT-SQL [7]	62.7	57.2
Light RAT-SQL[8]	-	60.25
Our model (T-RAT)	-	62.09

the same ($p=0.0047$) for Light RAT-SQL and T-RAT.

Table 3. Test Set Accuracy (and 95% confidence interval) of T-RAT, RAT-SQL, Light RAT-SQL

Model	Accuracy
T-RAT	61.58 ± 0.66
Light RAT-SQL	58.99 ± 1.04
RAT-SQL	58.90 ± 0.50

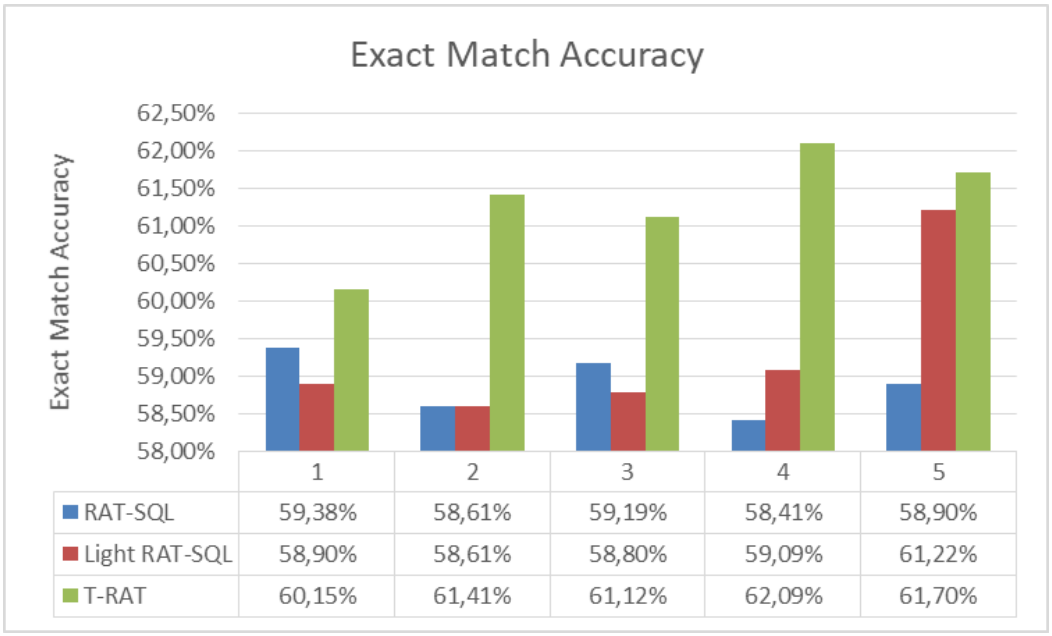

Figure 3. Exact Match Accuracy of RAT-SQL, Light RAT-SQL, and T-RAT

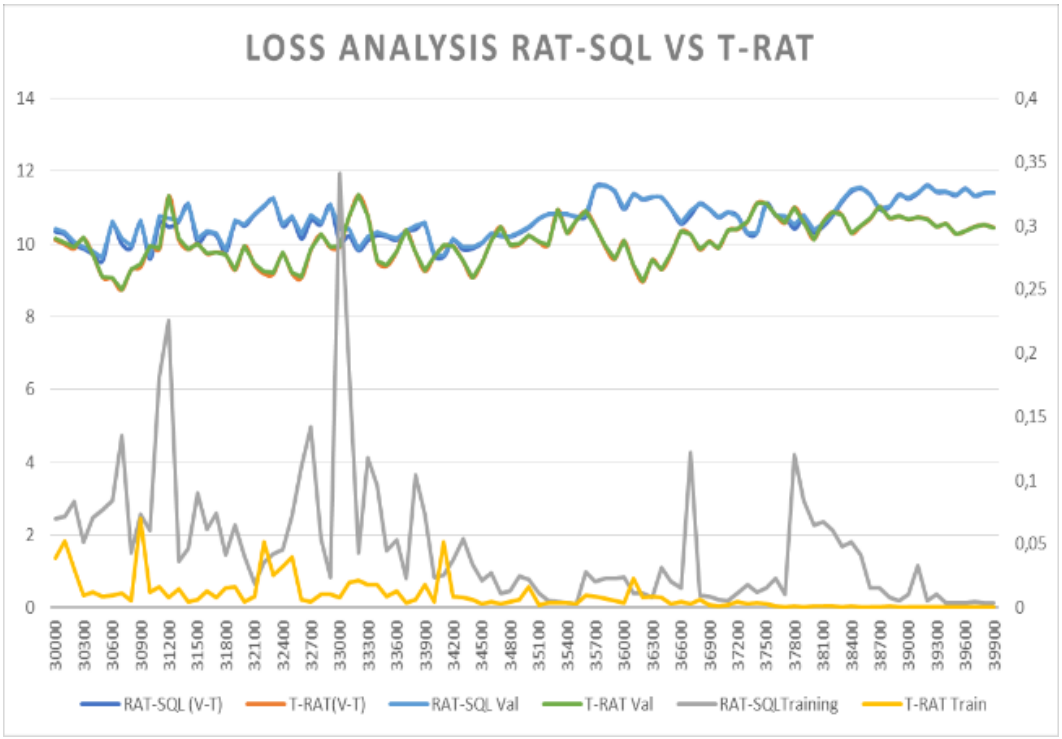

Figure 4. Loss Comparison between RAT-SQL and T-RAT

Discussion

Regularization

Topology-induced feature mixing, serving as a regularization mechanism, plays a pivotal role in mitigating overfitting within the model. By subjecting the model to an extensive array of patterns and relationships, this form of regularization diminishes the likelihood of the model memorizing the training data excessively. Instead, it fosters a propensity for the model to generalize adeptly when confronted with previously unseen data.

In Figure 4, the cross-validation loss of RAT-SQL and T-RAT is depicted (on the left y-axis). The discernible divergence in the cross-validation loss between the two models suggests that the incorporation of topology-induced adjustments in T-RAT has effectively curtailed overfitting tendencies, thereby promoting superior generalization compared to RAT-SQL. The nuanced alterations guided by the topology-induced regularization contribute to the model's enhanced capacity to extend its predictive capabilities beyond the confines of the training dataset.

Stability

Learning stability refers to the robustness and consistency of a machine learning model's performance across different training runs or datasets. A stable learning process implies that the model's behavior remains relatively consistent and reliable, even when subjected to variations in training data, initial conditions, or hyperparameter settings.

In scrutinizing the learning curves of the model over epochs or iterations, as presented in Figure 4 (corresponding to the right y-axis), it becomes evident that the T-RAT learning curve exhibits greater stability compared to RAT-SQL. The T-RAT model consistently converges across multiple instances of training runs, reflecting a heightened degree of reliability in its learning dynamics.

Alignment And Sensitivity

The graphical representation of the alignment generated by RAT-SQL, Light RAT-SQL, and T-RAT is illustrated in Figure 1. A nuanced examination of T-RAT reveals an enhanced sensitivity to Schema Linking, exemplified by the discernible manifestation of unwarranted attention between the "model_list" table and the token "model." This observation underscores the model's pronounced responsiveness to the intricacies of schema-linking relationships within the input structure. It is imperative to underscore that the manifestation of superfluous attention underscores the need to temper the schema-linking inductive bias[69]. As this bias is mitigated, an anticipatory improvement in the model's overall behavior is anticipated. Reducing the schema-linking inductive bias is thus posited as an essential strategy for enhancing the model's discernment and performance.

Conclusion

The Topological Relation Aware Transformer (T-RAT) is a specialized head transformer designed to address the problem of Text-To-SQL. T-RAT achieves this by leveraging the topology structure of pre-existing relations between input tokens and spreading each open set to one head of the Transformer. This specialization enables the model to capture diverse and contextually relevant relationship patterns within complex data, leading to improved accuracy. T-RAT also utilizes topology-induced feature mixing and an improved spreading algorithm to address the limitations of misalignment of tokens.

Experimental results demonstrate that T-RAT achieves higher exact match accuracy compared to baseline models RAT-SQL and Light RAT-SQL. The improved accuracy is attributed to the specialized head transformer and the effective utilization of pre-existing relations within the topology. It presents a promising approach to improving the accuracy

of text-to-SQL query generation models in low-resource settings.

Limitations

T-RAT-SQL was not enhanced with large language models such as GPT, BERT, or ELECTRA since this requires large GPU resources that we do have not. Enhancing our proposed model with other LLMs can be crucial to determining its effectiveness with competitive SOTA models.

Acknowledgments

We would like to thank Christian BOPE for his leadership talks and guidance that shaped this work. We also thank anonymous reviewers for their crucial input.

Conflict Of Interest Statement

All authors declare that they have no conflicts of interest.

References

[1] T. Scholak, R. Li, D. Bahdanau, H. de Vries, and C. Pal, "DuoRAT: Towards Simpler Text-to-SQL Models," Oct. 2020, doi: 10.18653/v1/2021.naacl-main.103.

[2] W. Hou and Y. Nie, "Seq2seq-Attention Question Answering Model.

[3] O. Goldman, V. Latcinnik, U. Naveh, A. Globerson, and J. Berant, "Weakly-supervised Semantic Parsing with Abstract Examples," Nov. 2017, [Online]. Available: http://arxiv.org/abs/1711.05240.

[4] X. V. Lin, R. Socher, and C. Xiong, "Bridging Textual and Tabular Data for Cross-Domain Text-to-SQL Semantic Parsing," Dec. 2020, [Online]. Available: http://arxiv.org/abs/2012.12627

[5] I Gur, S. Yavuz, Y. Su, and X. Yan, "DialSQL: Dialogue Based Structured Query Generation."

[6] X. Xu, C. Liu, and D. Song, "SQLNet: Generating Structured Queries from Natural Language Without Reinforcement Learning," Nov. 2017, [Online]. Available: http://arxiv.org/abs/1711.04436.

[7] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, "RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers," 2020. [Online]. Available: https://github.com/Microsoft/rat-sql.

[8] N. M. Ndongala, "Light RAT-SQL: A RAT-SQL with More Abstraction and Less Embedding of Pre-existing Relations," Texila Int. J. Acad. Res., vol. 10, no. 2, pp. 1–11, 2023, doi: 10.21522/tijar.2014.10.02.art001.

[9] G. Huilin, G. Tong, W. Fan, and M. Chao, "Bidirectional attention for SQL generation," in 2019 IEEE 4th International Conference on Cloud Computing and Big Data Analytics, ICCCBDA 2019, Institute of Electrical and Electronics Engineers Inc., Apr. 2019, pp. 676–682. doi: 10.1109/ICCCBDA.2019.8725626.

[10] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators," Mar. 2020, [Online]. Available: http://arxiv.org/abs/2003.10555.

[11] M. Lewis et al., "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.13461.

[12] M. Shoeybi, M. Patwary, R. Puri, P. Legresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism," 2020. [Online]. Available: https://github.com/.

[13] T. B. Brown et al., "Language Models are Few-Shot Learners," 2020. [Online]. Available: https://commoncrawl.org/the-data/.

[14] Z. Lan et al., "ALBERT: A Lite Bert for Self-Supervised Learning Of Language Representations," 2020. [Online]. Available: https://github.com/google-research/ALBERT.

[15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," 2019. [Online]. Available: https://github.com/codelucas/newspaper.

[16] V. Zhong, C. Xiong, and R. Socher, “Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning,” Aug. 2017, [Online]. Available: http://arxiv.org/abs/1709.00103.

[17] A. Vaswani et al., “Attention Is All You Need,” Jun. 2017, [Online]. Available: http://arxiv.org/abs/1706.03762.

[18] O. Vinyals, M. Fortunato, and N. Jaitly, “Pointer Networks,” Jun. 2015, [Online]. Available: http://arxiv.org/abs/1506.03134.

[19] Z. Tu, Z. Lu, L. Yang, X. Liu, and H. Li, “Modeling coverage for neural machine translation,” in 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers, Jan. 2016, pp. 76–85. doi: 10.18653/v1/p16-1008.

[20] P. Shaw, J. Uszkoreit, G. Brain, and A. Vaswani, “Self-Attention with Relative Position Representations,” 2018.

[21] L. Zehui, P. Liu, L. Huang, J. Chen, X. Qiu, and X. Huang, “DropAttention: A Regularization Method for Fully Connected Self-Attention Networks,” Jul. 2019, Accessed: Apr. 04, 2022. [Online]. Available: http://arxiv.org/abs/1907.11065.

[22] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, “On the dangers of stochastic parrots: Can language models be too big?” in FAccT 2021 - Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, Association for Computing Machinery, Inc, Mar. 2021, pp. 610–623. doi: 10.1145/3442188.3445922.

[23] D. E. B. A. D. Ecoding and E. Bert, “Entangled A Ttention with D Is -,” 2021.

[24] W. Fedus, B. Zoph, and N. Shazeer, “Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity,” 2022.

[25] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Bidirectional Encoder Representations from Transformers),” Bert-Ppt, 2018.

[26] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,” Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.01108.

[27] Y. Liu et al., “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” Jul. 2019, [Online]. Available: http://arxiv.org/abs/1907.11692.

[28] H. Li, J. Zhang, C. Li, and H. Chen, “RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL,” Feb. 2023, [Online]. Available: http://arxiv.org/abs/2302.05965

[29] L. Zhao, H. Cao, and Y. Zhao, “GP: Context-free Grammar Pre-training for Text-to-SQL Parsers,” Jan. 2021, [Online]. Available: http://arxiv.org/abs/2101.09901.

[30] P. Shi et al., “Learning Contextual Representations for Semantic Parsing with Generation-Augmented Pre-Training,” Dec. 2020, [Online]. Available: http://arxiv.org/abs/2012.10309

[31] T. Yu et al., “GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing,” Sep. 2020, [Online]. Available: http://arxiv.org/abs/2009.13845.

[32] X. Deng, A. H. Awadallah, C. Meek, O. Polozov, H. Sun, and M. Richardson, “Structure-Grounded Pretraining for Text-to-SQL,” Oct. 2020, doi: 10.18653/v1/2021.naacl-main.105.

[33] P. Yin and G. Neubig, “TRANX: A Transition-based Neural Abstract Syntax Parser for Semantic Parsing and Code Generation,” Oct. 2018, [Online]. Available: http://arxiv.org/abs/1810.02720.

[34] P. Yin, C. Zhou, J. He, and G. Neubig, “STRUCTVAE: Tree-structured Latent Variable Models for Semi-supervised Semantic Parsing.” [Online]. Available: http://pcyin.me/struct.

[35] L. Dong and M. Lapata, “Language to Logical Form with Neural Attention,” Jan. 2016, [Online]. Available: http://arxiv.org/abs/1601.01280.

[36] L. Dong and M. Lapata, “Coarse-to-Fine Decoding for Neural Semantic Parsing,” May 2018, [Online]. Available: http://arxiv.org/abs/1805.04793.

[37] A. Gopalan et al., “Neural Structured Learning: Training Neural Networks with Structured Signals,” in WSDM 2021 - Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021. doi: 10.1145/3437963.3441666.

[38] I. Gopalan et al., “Neural Structured Learning,” 2020. doi: 10.1145/3394486.3406701.
[39] II. Bogin, M. Gardner, and J. Berant, “Global Reasoning over Database Structures for Text-to-SQL Parsing,” 2019.
[40] Y. Ma and J. Tang, “Graph Neural Networks in Natural Language Processing,” in Deep Learning on Graphs, 2021. doi: 10.1017/9781108924184.015.
[41] I. Hui et al., “S^{2}SQL: Injecting Syntax to Question-Schema Interaction Graph Encoder for Text-to-SQL Parsers,” Mar. 2022, [Online]. Available: http://arxiv.org/abs/2203.06958.
[42] R. Cai, J. Yuan, B. Xu, and Z. Hao, “SADGA: Structure-Aware Dual Graph Aggregation Network for Text-to-SQL,” Oct. 2021, [Online]. Available: http://arxiv.org/abs/2111.00653.
[43] R. Cao, L. Chen, Z. Chen, Y. Zhao, S. Zhu, and K. Yu, “LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations,” Jun. 2021, [Online]. Available: http://arxiv.org/abs/2106.01093.
[44] OpenAI et al., “GPT-4 Technical Report,” vol. 4, pp. 1–100, 2023, [Online]. Available: http://arxiv.org/abs/2303.08774.
[45] M. Pourreza and D. Rafiei, “DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction,” no. NeurIPS, pp. 1–34, 2023, [Online]. Available: http://arxiv.org/abs/2304.11015.
[46] I. Gao et al., “Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation,” 2023, [Online]. Available: http://arxiv.org/abs/2308.15363.
[47] I. A. Dahl et al., “EXPANDING THE SCOPE OF THE ATIS TASK: THE ATIS-3 CORPUS.”
[48] Y. Gan et al., “Towards robustness of text-to-SQL models against synonym substitution,” ACL-IJCNLP 2021 - 59th Annu. Meet. Assoc. Comput. Linguist. 11th Int. Jt. Conf. Nat. Lang. Process. Proc. Conf., pp. 2505–2515, 2021, doi: 10.18653/v1/2021.acl-long.195.
[49] P. Utama et al., “An End-to-end Neural Natural Language Interface for Databases,” 2018, [Online]. Available: http://arxiv.org/abs/1804.00401.
[50] T. Yu et al., “Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task,” Sep. 2018, [Online]. Available: http://arxiv.org/abs/1809.08887.
[51] T. Yu et al., “SPARC: Cross-domain semantic parsing in context,” ACL 2019 - 57th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf., pp. 4511–4523, 2020, doi: 10.18653/v1/p19-1443.
[52] X. Yu et al., “Dataset and enhanced model for eligibility criteria-to-SQL semantic parsing,” Lr. 2020 - 12th Int. Conf. Lang. Resour. Eval. Conf. Proc., no. May, pp. 5829–5837, 2020.
[53] H. Zhang et al., “CSS: A Large-scale Cross-schema Chinese Text-to-SQL Medical Dataset,” Proc. Annu. Meet. Assoc. Comput. Linguist., pp. 6970–6983, 2023, doi: 10.18653/v1/2023.findings-acl.435.
[54] Y. Gan, X. Chen, and M. Purver, “Exploring Underexplored Limitations of Cross-Domain Text-to-SQL Generalization,” EMNLP 2021 - 2021 Conf. Empir. Methods Nat. Lang. Process. Proc., pp. 8926–8931, 2021, doi: 10.18653/v1/2021.emnlp-main.702.
[55] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington, “The ATIS Spoken Language Systems Pilot Corpus.”
[56] Q. Min, Y. Shi, and Y. Zhang, “A pilot study for Chinese SQL semantic parsing,” EMNLP-IJCNLP 2019 - 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Jt. Conf. Nat. Lang. Process. Proc. Conf., pp. 3652–3658, 2019, doi: 10.18653/v1/d19-1377.
[57] D. Sean and P. S. Meltzer, “GEOquery: A bridge between the Gene Expression Omnibus (GEO) and BioConductor,” Bioinformatics, vol. 23, no. 14, pp. 1846–1847, Jul. 2007, doi: 10.1093/bioinformatics/btm254.
[58] T. Shi, C. Zhao, J. Boyd-Graber, H. Daumé, and L. Lee, “On the potential of lexico-logical alignments for semantic parsing to SQL queries,” Find. Assoc. Comput. Linguist. Find. ACL EMNLP 2020, pp. 1849–1864, 2020, doi: 10.18653/v1/2020.findings-emnlp.167.
[59] M. Singh et al., “CL Scholar: The ACL Anthology Knowledge Graph Miner,” NAACL HLT 2018 - 2018 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. Proc.

Demonstr. Sess., pp. 16–20, 2018, doi: 10.18653/v1/n18-5004.
[60] A. Suhr, M. W. Chang, P. Shaw, and K. Lee, "Exploring unexplored generalization challenges for cross-database semantic parsing," Proc. Annu. Meet. Assoc. Comput. Linguist., pp. 8372–8388, 2020, doi: 10.18653/v1/2020.acl-main.742.
[61] L. R. Tang and R. J. Mooney, "A u t o m a t e d Construction of Database Interfaces: Integrating Statistical and Relational Learning for Semantic Parsing," 1996.
[62] J. MUNKRES, Topology. Pearson College Div, 2000. [Online]. Available: https://www.amazon.com/Topology-2nd-James-Munkres/dp/0131816292.
[63] .P Yin and G. Neubig, "A Syntactic Neural Model for General-Purpose Code Generation," Apr. 2017, [Online]. Available: http://arxiv.org/abs/1704.01696.
[64] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. Mcclosky, "The Stanford CoreNLP Natural Language Processing Toolkit," 2014.
[65] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," 2014. [Online]. Available: http://nlp.
[66] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Comput., vol. 9, no. 8, 1997, doi: 10.1162/neco.1997.9.8.1735.
[67] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," 2019.
[68] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980.
[69] J. Guo et al., "Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation," 2019.
[70] B. L. Edelman, S. Goel, S. Kakade, and C. Zhang, "Inductive Biases and Variable Creation in Self-Attention Mechanisms," Proc. Mach. Learn. Res., vol. 162, pp. 5793–5831, 2022.