TEMPSQL: TEMPLATE BASED TEXT-TO-SQL GENERATION ON SINGLE SOURCE DATABASE

Abhishek Bhatt [CS18MDS11018], Anand Kumar Rai [CS18MDS11022]

Department of Computer Science & Engineering, IITH

Project Guide:
Dr. Maunendra S Desarkar
Asst. Prof ,
Dept. of CSE, IITH

Acknowledgement

We would like acknowledge the contributors of Open-Source packages/libraries like NLTK, PyDictionary etc. and framework like Tensorflow which have facilitated the implementation of our approach.

We would also like to thank the researchers of our problem domain Text-to-SQL , whose literature provided us the necessary insights and knowledge required to tackle the challenges during the execution of the project.

And last but not the least, we would like to thank our Project Guide Dr. Maunendra, without whose constant guidance, support, feedback and rich contributions , the project would not have been successfully completed within stipulated time.

Table of Contents

Abstract

The quest for knowledge is deeply human. With relational databases holding abundance of information which is useful for both academia and industry, it is quite natural to see growing usage and utility of Structured Query Language (SQL) for knowledge extraction. But inspite of SQL being very high level language with simple English commands, its never within comfortable reach of non-technical users and hence a lot of research efforts have been done so far in Natural Language(NL) to SQL translation. But the flip side of all the endeavors made in this area is that we are still able to only achieve around 72% maximum accuracy and the solutions are not very industry friendly. In this paper, we are attempting to provide the industry who are using RDBMS with menu-based reporting tool on top of it with a simple user-interface where they can easily interact with their data. For achieving the same, we are proposing TempSQL, which is a template-based NL to SQL model on single source database, where training and test will take place on same database. The main idea behind TempSQL is synthetic data generation from existing NL to SQL templates, which is used to train an attention based Bi-Directional SEQ2SEQ model for achieving around 70% accuracy on multi-table databases from SPIDER [1] dataset.

Keywords: SQL, NLP, Bi-Directional SEQ2SEQ model, Attention, SPIDER

The implementation of our approach is hosted at:

https://github.com/AbhishekBhattGitHub/TempSQL

Introduction

Problem Statement

As the industry reserve of data kept increasing in the past two decades, data storage formats keep evolving from structured to un-structured data. But still the majority of data is kept and managed in Structured Tabular Format in relational database and queried through SQL.

As SQL largely remained a developer or DBA language since its inception, the non-technical users or top-tier management of any company has to rely on the business analytics software like Tableau, Power BI or their inhouse software to gain insights from their data repository.

These analytical tools succeeded in abstracting the SQL complexity from the users by hard coding the queries based on input provided by the users. However, they do not support on-demand natural language querying which has been a highlight feature for the querying systems of unstructured data such as the web search engines in the past couple of decades.

A lot of advancements in query understanding and information retrieval techniques has been made possible through AI/ML research specifically in NLP domain. But the research results of Text-to-SQL conversion are yet to be successfully commercialized because state-of-the art neural models like RAT-SQL [2], RYAL-SQL [3] etc. on the SPIDER dataset yields only 60–65 exact matching accuracy.

Our Approach

In this project, we have attempted to develop an end-to-end commercial solution for Text-to-SQL problem which can be easily integrated with existing systems in the industry. Our proposed solution consists of two modules:

a) **TempSQL:** An attention based Bidirectional Seq2Seq Model trained on synthetic data generated from templates

b) **Talk2DB:** An interactive interface for inference where user can leverage autocomplete feature based on Language Modelling on Questions dataset and get desired results

Both of the above modules will be discussed in details in coming sections of the report. We have tested our solution on two diverse multi-table databases of SPIDER dataset viz. college_2 and store_1 and got the test accuracy of 72% and 81% respectively.

Related Work

Natural language query understanding for unstructured textual sources has seen significant progress over the last couple of decades due to the continued research efforts in this area.

The timeline below highlights the transition in Text-to SQL generation approach in last 2-3 decades and some important models based on those approaches.

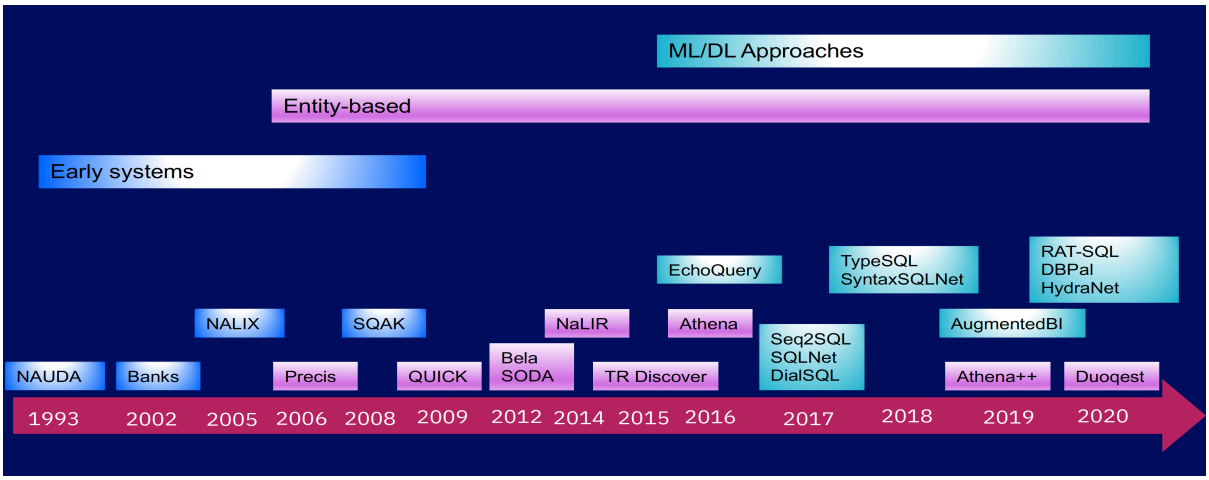


Figure 1: State of the Art and Open Challenges in NL Interfaces to Data, Ozcan et al , IBM, SIGMOD 2020[4]

With the advent of attention based embeddings in NLP [5] , the semantic understanding of the text has increased manifold and its impact on Text-to SQL problem is also visible. Now more and more solutions are DL based but still the one which is state-of-art is a hybrid of DL & Entity based approach.

It is also to be noted that before the publication of SPIDER dataset in 2018 which is a collection of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains, most of the models were benchmarked on the WIKISQL[6] dataset of 80,654 questions and simple SQL queries on single table database.

Hence , the current models has more applicability on the real world systems. Below graph shows the current benchmark trend on the SPIDER dataset.

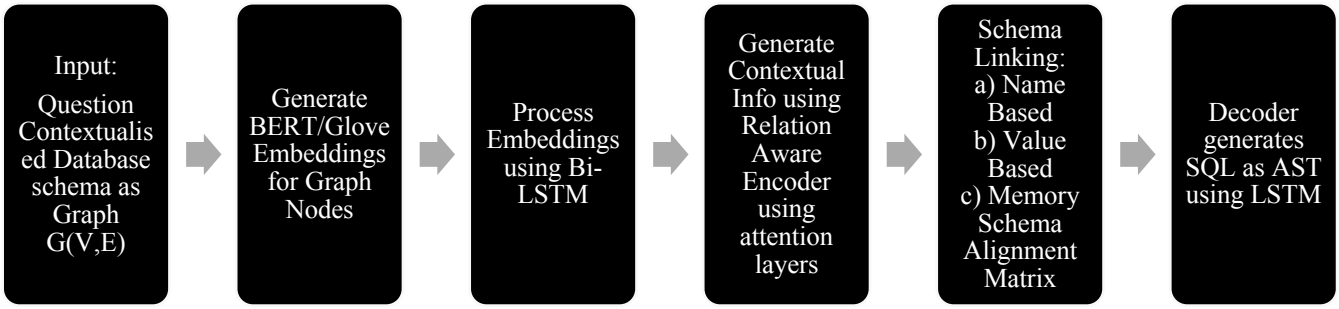
Figure 2: Benchmark results on Text-to SQL on SPIDER dataset taken from paperswithcode.com

We have reviewed two recent research papers in this domain, RAT-SQL & PHOTON [7].

RAT-SQL

As the name RAT-SQL (Relation-Aware-Transformer) suggests, it is an attention-based Sequence to Sequence Model where decoder is aware of the table relationships and table schema in addition to the contextual info generated from encoder.

The methodology summary has been explained below:



Schema Linking is entity-based technique which used for column alignment in NL & SQL. Without this the model has reported an accuracy of 40% which way lower than the overall accuracy of the model which 62.7%. The detailed results are given below:

Model	Dev	Test
IRNet (Guo et al., 2019)	53.2	46.7
Global-GNN (Bogin et al., 2019b)	52.7	47.4
IRNet V2 (Guo et al., 2019)	55.4	48.5
RAT-SQL (ours)	**62.7**	**57.2**
With BERT:		
EditSQL + BERT (Zhang et al., 2019)	57.6	53.4
GNN + Bertrand-DR (Kelkar et al., 2020)	57.9	54.6
IRNet V2 + BERT (Guo et al., 2019)	63.9	55.0
RYANSQL V2 + BERT (Choi et al., 2020)	**70.6**	60.6
RAT-SQL + BERT (ours)	69.7	**65.6**

Table 2: Accuracy on the Spider development and test sets, compared to the other approaches at the top of the dataset leaderboard as of May 1st, 2020. The test set results were scored using the Spider evaluation server.

Split	Easy	Medium	Hard	Extra Hard	All
RAT-SQL					
Dev	80.4	63.9	55.7	40.6	62.7
Test	74.8	60.7	53.6	31.5	57.2
RAT-SQL + BERT					
Dev	86.4	73.6	62.1	42.9	69.7
Test	83.0	71.3	58.3	38.4	65.6

Table 3: Accuracy on the Spider development and test sets, by difficulty as defined by Yu et al. (2018b).

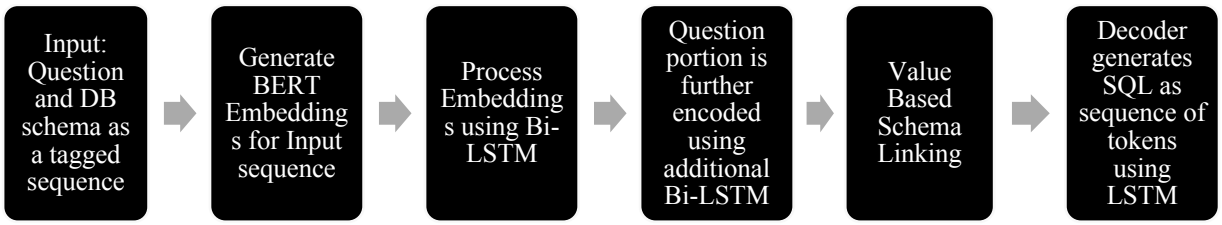
Model	Accuracy (%)
RAT-SQL + value-based linking	**60.54 ± 0.80**
RAT-SQL	55.13 ± 0.84
w/o schema linking relations	40.37 ± 2.32
w/o schema graph relations	35.59 ± 0.85

Figure 3: Results of RAT-SQL Model on SPIDER dataset

PHOTON

Photon also has the same approach as of RAT-SQL but the input is a combined sequence of Database Schema and Question instead of GRAPH as in RAT-SQL. Similarly, output is sequence of SQL tokens instead of an AST as in RAT-SQL.

The approach has been briefly described below:



An inference model with human-in-the-loop has also been developed and deployed as *naturalsql.com*. The purpose of the interface is correctness checking and the question corrector for ambiguous & untranslatable questions using user feedback are additional steps taken for improving performance.

The results of the model are summarized below:

Model	EM Acc.
GNN (Bogin et al., 2019a)	40.7
Global-GNN (Bogin et al., 2019b)	52.7
EditSQL + BERT (Zhang et al., 2019)	57.6
GNN+Bertrand-DR† (Kelkar et al., 2020)	57.9
EditSQL+Bertrand-DR† (Kelkar et al., 2020)	58.5
IRNet + BERT (Guo et al., 2019)	61.9
RYANSQL + BERT † (Choi et al., 2020)	66.6
PHOTON	63.2

† denotes unpublished work on arXiv.

Table 3: Experimental results on the Spider Dev set (%). EM Acc. denotes the exact set match accuracy.

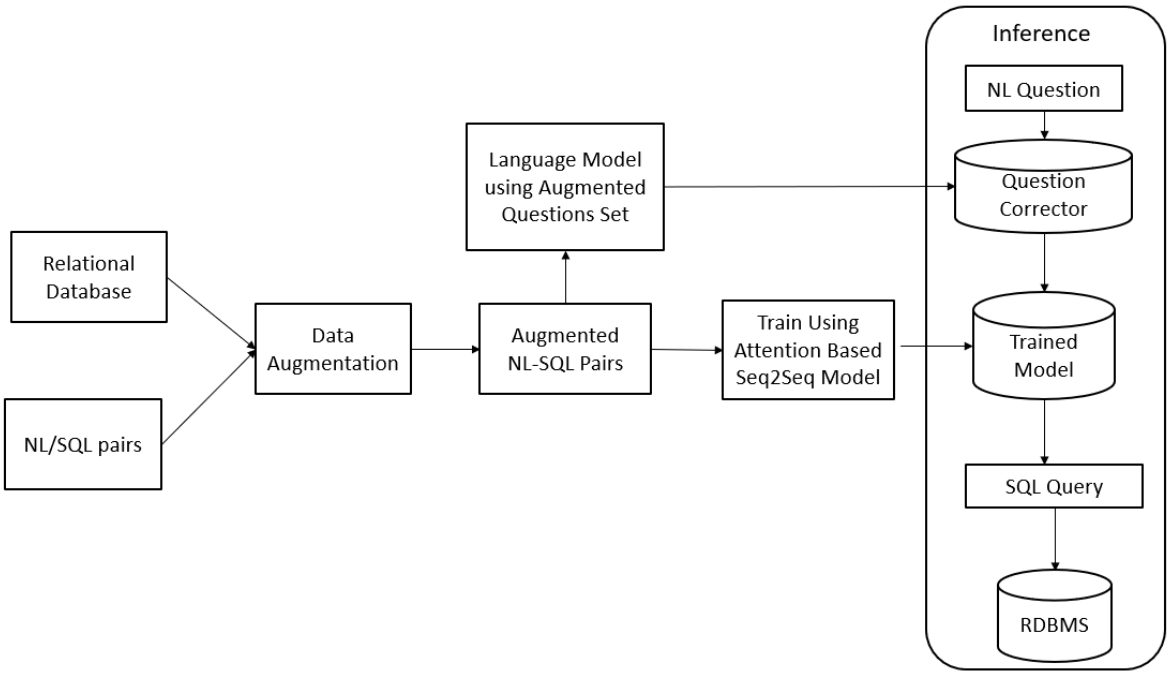
Figure 5: Results of PHOTON & Contemporary Models on SPIDER dataset

TEMPSQL: Template based Text-to-SQL Generation on Single Source Database

In the upcoming sections, we will brief about the System Design, Experimental Setup & Results and we will also touch upon the aspects for future improvements

System Design

System Architecture



The Architecture can be primarily divided in to following sections:

- Data Preparation
- Model for NL-SQL
- Assistive user interface using Language model

The Relational database and NL-SQL pairs are being used from Spider dataset; however, the small count of NL-SQL pairs does not suffice for the Deep learning purpose. Therefore, Data augmentation techniques have been used to create synthetic data. This process is partly automated, small level of manual intervention is needed for creating templates and cleaning the synonyms.

This data is then used for creating two Models, a Seq2Seq model for translating NL queries to SQL queries and a Language model that learns the NL queries and assists the user during inference.

A web-based inference UI has been created that takes NL queries as input and then outputs the respective SQL query and results from the database. While the user is typing the NL query, they

are assisted by the Auto complete feature which predicts the next word and suggests top 3 most probable words that can possibly help the user to complete the query.

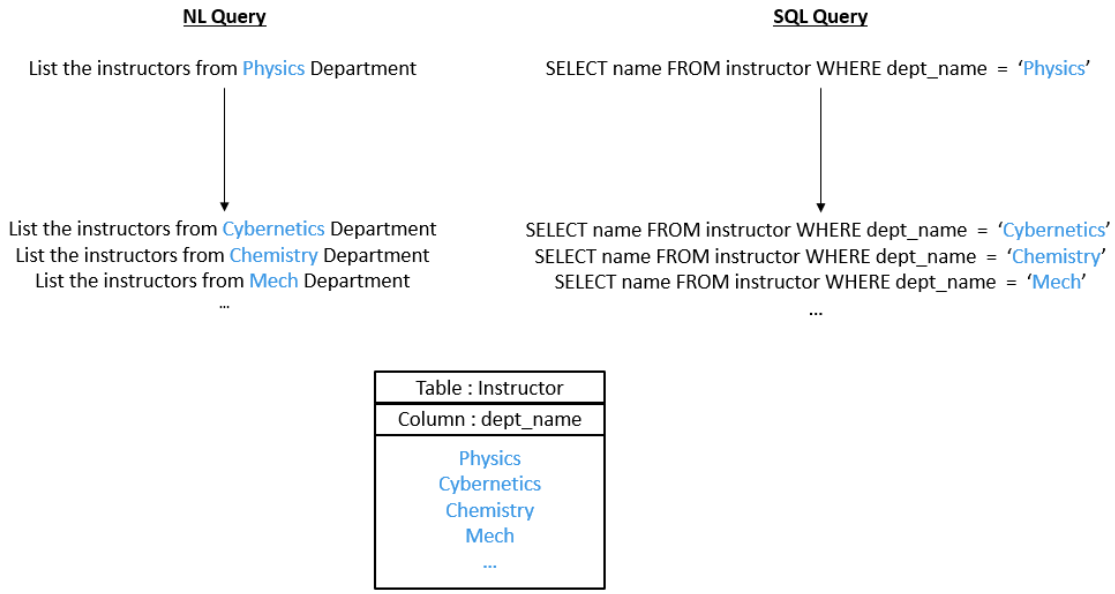
Data Preparation

The number of NL-SQL pairs in College_2 database is 170 which is not enough to train the model therefore data augmentation is required.
Following data augmentation techniques were used:

- Slot Filling
- Synonym based word augmentation

Slot Filling

In this technique the values present in the NL & SQL queries are replaced with other values present in the Database. Here is an example of the NL query where the "Physics" Department name is being replaced with other unique Department names from the Database. Similarly, the SQL query is also augmented as shown below:



In order to achieve this, a Template of NL and SQL queries is created using the existing NL-SQL queries present in the College_2 dataset.

For creation of templates from SPIDER dataset NL-SQL pairs , we have parsed the SQL query and identified the table name , column name & the column value that will be replaced with a placeholder for slot-filling.

Template Generation for College_2 database		
	Template	**NL-SQL pair**
Manual	22 (28%)	5431 (37%)
Automatic	57 (72%)	9175 (63%)
Total	79	14516

Here is how the template and the respective NL & SQL augmented data for the earlier example look like:

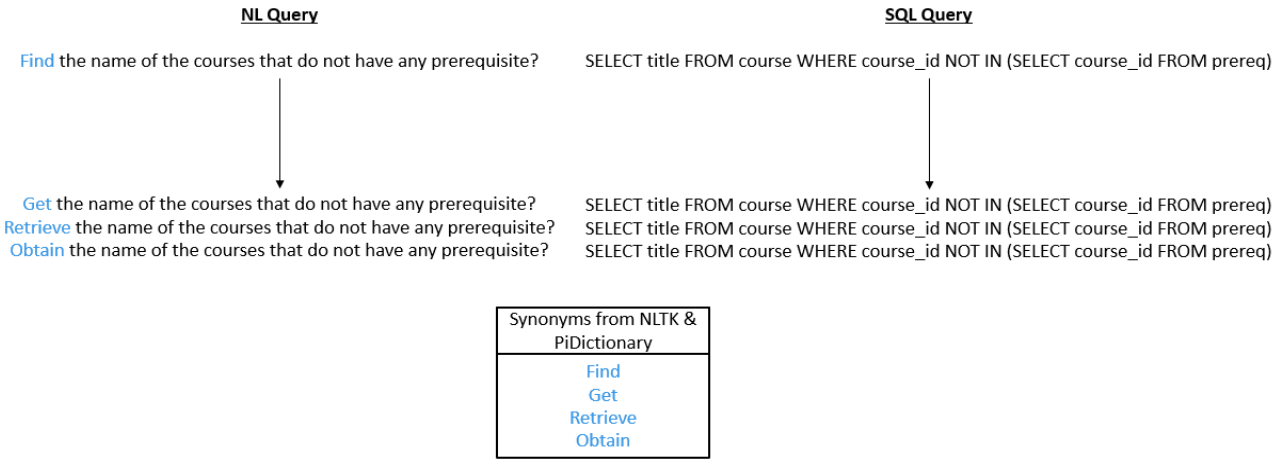
NL Template	SQL Template
List the instructors from {instructor.dept_name} department?	SELECT name FROM instructor WHERE dept_name = '{instructor.dept_name}'

Automatically generated NL queries	Automatically generated SQL queries
List the instructors from Physics Department	SELECT name FROM instructor WHERE dept_name = 'Physics'
List the instructors from Cybernetics Department	SELECT name FROM instructor WHERE dept_name = 'Cybernetics'
List the instructors from Chemistry Department	SELECT name FROM instructor WHERE dept_name = 'Chemistry'
List the instructors from Mech Department	SELECT name FROM instructor WHERE dept_name = 'Mech'
...	...
...	

The attribute is marked inside curly braces {}, these attributes are in the format of {TableName.ColumnName}. During slot filling process this attribute is replaced with all unique values present under the respective Table / Column from the database. In the above example the table name is "instructor", column name is "dept_name" and "Physics", "Cybernetics", "Chemistry" and "Mech" are the unique values present under the "dept_name" column.

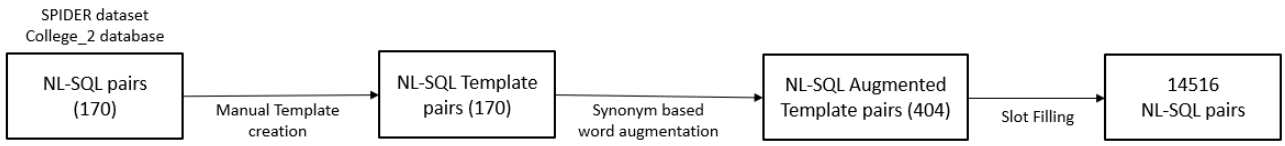
Synonym based word augmentation

In this technique we find the unique words present in all the NL Template queries and then find their synonyms using NLTK and PyDictionary libraries. These synonyms are manually cleaned as not all the synonyms thus found are accurate or fit the application.
Finally, these words are replaced in the NL Template queries with their respective synonyms. The respective SQL query remains unchanged.



Data Augmentation Results

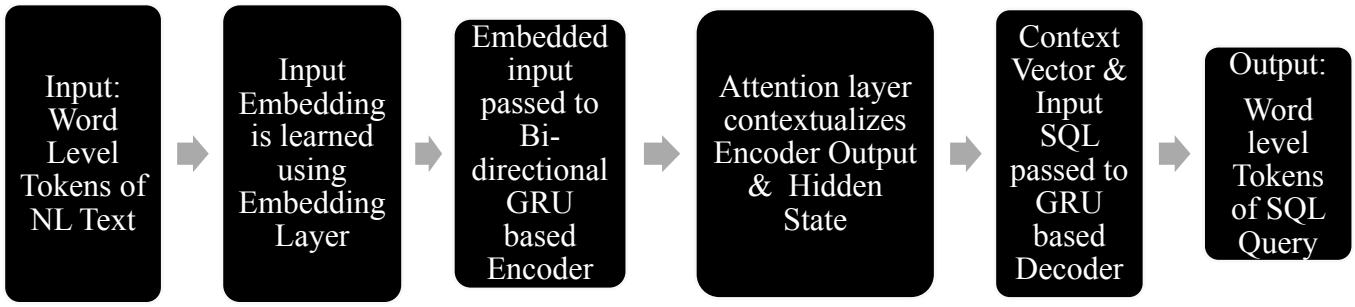
The "College_2" database in the Spider dataset contains around 170 numbers of NL-SQL query pairs. Each of these NL-SQL pair is manually checked if it can be augmented using slot filling. Below is an example which can be augmented using slot filling.
"how many rooms does the chandler building have?"
Here "chandler" is the name of the building and we can slot fill it with other building names from the database.
Below is an example which cannot be augmented using slot filling:
"Find the name and building of the department with the highest budget."
The template thus created is then augmented by replacing the words with their synonyms. Finally, this augmented template is used to perform the slot filling. Below are the results of the same:



		Count of NL-SQL pairs	
Dataset	**Database**	**Before augmentation**	**After augmentation**
Spider	College_2	170	14516
Spider	Store_1	112	35672

TempSQL Model

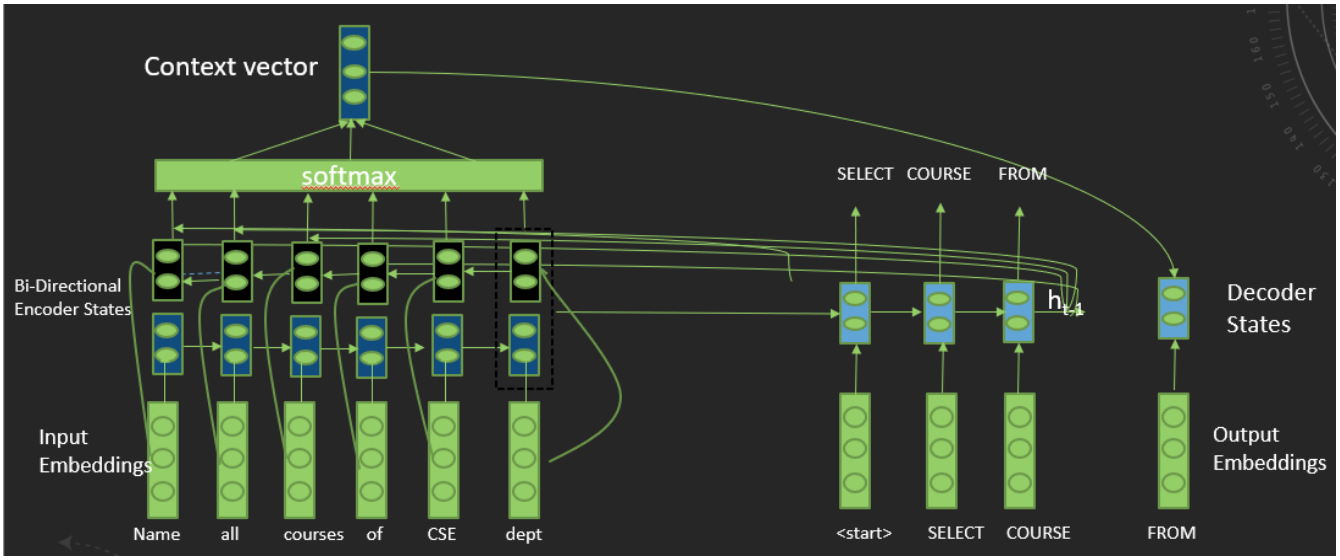
The model used for training on synthetic generated data in TempSQL is Bidirectional Seq2Seq model with additive Attention[8]. The model can be summarized with the following sequential diagram.



The pseudo code of the implemented model is as follows:

```
X= Embedding(X)                                                    [NL]
EO, EH = Encoder(X)
attention_score      =FC(tanh(FC(EO) + FC(EH)))                    [Badhnau Attention]
context_vector    = sum( softmax(attention_score ) * EO)
Y = Embedding(Y)                                                   [SQL]
output     = Decoder(concat(Y, context vector))
```

The above detailed sequence is detailed schematically in the below diagram



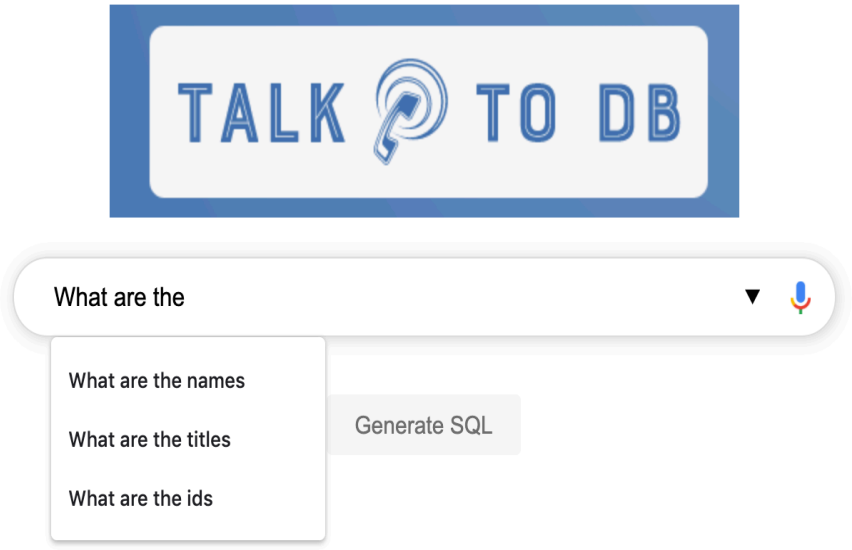
Assistive Inference UI

TalkToDB is a web based assistive UI that utilizes Language model to auto complete the query being entered by the user. The Language model is created using the NL queries present in the Training data.

The Auto complete feature predicts the next word and suggests top 3 most probable words that can possibly help the user to complete the query. The user can either choose from the options or enter any other word as per their wish.

Once the user submits the NL query then it is passed to the Seq2Seq model which returns the translated SQL query. At this state the user has an option to modify the SQL query if needed. Once the user it satisfied with SQL query and presses the “Fetch Result” button, then the result is fetched from the database by running the SQL query and the results are shown on the Web UI.

The snapshot TalkToDB interface deployed at local server is shown below:



What are the names of all instructors in cybernetics department

Generate SQL

SELECT name FROM instructor WHERE dept_name = 'Cybernetics'

Fetch Results

name
McKinnon
Pimenta
Bietzk
Dale

Evaluation

Experimental Setup

We have used the TensorFlow implementation [9] of Seq2Seq Attention based model. We have tried various variants of the model like training the model without attention, training with its multi-head attention and using bidirectional encoder instead of the simple encoder.

The dataset used for training & test for above mentioned variants is <NL, SQL> pairs corresponding to college_2 database of SPIDER dataset. From the given dataset we have manually curated the templates and used the synthetic dataset created from data augmentation methodology explained above. We have validated our model on a different database i.e Store_1 of SPIDER dataset and its result are tabulated in below section.

We have taken exact match accuracy as an evaluation metric for this task as any deviation in generated SQL will not give desired result on being executed on the database.

After observing BiDirectional Seq2Seq with Attention model outperforms our other attempts, we have done parameter tuning like trying various batch sizes, varying embedding size, no. of hidden states in encoder & decoder and optimizer learning rate for better accuracy measure and reduce variance.

The model parameters finalized after tuning are as below:

Encoder Type	**Bidirectional**
Encoder/Decoder Unit	GRU
Embedding Size	256
No. of Hidden Units	512
Batch Size	64
Attention Type	Bahdanau (Additive)
Optimizer	Adam
Learning Rate	0.001

Table 1: Model Parameters of the TempSQL model used for training

Results

The dataset of 16000 <NL, SQL> pairs is split in 80:20 ratio for train & test and results on various models tried by us after 10 Epochs are tabulated below:

Model (Dataset:college_2)	Train Accuracy	Test Accuracy
Vanilla Seq2Seq	27%	6%
Seq2Seq with Attention	62%	59%
Seq2Seq with Multi-Head Attention (heads=2)	49%	48%
Bidirectional Seq2Seq with Attention	73%	72%

Table 2: Train & Test results of various model of college_2 database

The validation of TempSQL algorithm is done on Store_1 database of SPIDER dataset having 110 <NL,SQL> pairs which has been augmented to 35672 pairs and the result has been tabulated below:

Model (Dataset: Store_1)	Train Accuracy	Test Accuracy
Bidirectional Seq2Seq with Attention	81%	79%

Table 3: Train & Test results of TempSQL model of store_1 database

For better understanding of the results, breakdown of accuracy of our model on different type of queries on college_2 & store_1 dataset is shown in below tables:

Types of Data (Dataset:college_2)	Samples (%)	Test Accuracy (%)
Easy (Simple No Rel Operator)	4%	87.9
Medium (Rel Op Present)	38%	99.7
Hard (Nested Queries)	19%	95.8
Difficult (Multiple Joins, INTERSECT, EXCEPT)	39%	29.7

Table 4: Test results of various type of queries on college_2 database

Types of Data (Dataset: Store_1)	Samples (%)	Test Accuracy (%)
Easy (Simple No Rel Operator)	48%	98%
Medium (Rel Op Present)	8%	90%
Hard (Nested Queries)	0%	NA
Difficult (Multiple Joins, INTERSECT, EXCEPT)	44%	58%

Table 5: Train & Test results of TempSQL model of store_1 database

Error Analysis

As it is evident in Table 4 & Table 5, the model does not perform well on difficult queries having either multiple join or INTERSECT, UNION or EXCEPT queries. On further investigation of generated SQL queries from test dataset it was found accuracy on Multiple Join, INTERSECT, UNION and EXCEPT queries are 96%, 0%, 0% and 0% respectively.
This clearly indicates that the model is facing issues in learning queries with above-mentioned queries due to which there is a dip in accuracy in dataset containing numerous INTERSECT, UNION and EXCEPT based queries. Below the attention graph of a incorrectly translated query:

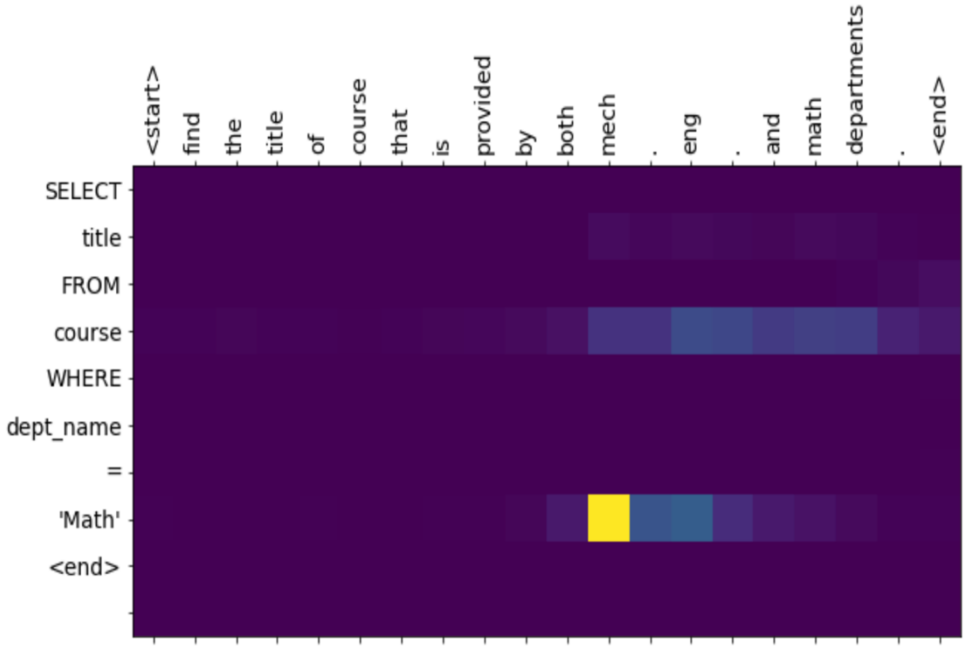


Figure 6 : Attention plot illustrating incorrect alignment of values (Math & mech. Eng.) in NL & SQL query

Conclusion

As clearly mentioned in the Introduction Section also, our solution is directed towards descriptive analytics requirement of the industry rather than improving state of art in Text-to SQL translation domain. The end-to-end package of TempSQL and Talk2DB provides a realistic template of the commercial solution to come for this task in very near future.

The features like self-learned embeddings, specialized training on the given database will allow the model to learn any domain specific database effectively and generate results with accuracy. At the same time the simplistic model design will provide flexibility to try different variants of model and robustness to the model by adapting to the new data by adding templates.

There are still some improvements that could be made for making the product to meet industry expectations and trust. The first and foremost is definitely accuracy for complex queries so that the user don't have to correct generated SQL repeatedly. Addressing the cold-start problem as this model is as good as the initial templates used and in scarcity of the templates, a lot of manual efforts to be done for generating the same. Making the model self-learning by learning from the incorrectly generated SQL during inference will also be a great step towards improvement. The Talk2DB can also be made more interactive by generating graphs for the data queried and by making the interaction a conversational one. Application of our solution to other forms of structured knowledge (e.g., graphs) is another interesting direction for NLP research.

References

1. 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. 2018b. Spider: A largescale 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. Association for Computational Linguistics.

2. 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. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7567–7578, Online. Association for Computational Linguistics.

3. DongHyun Choi, Myeong Cheol Shin, Eung Gyun Kim, and Dong Ryeol Shin. 2020. RYANSQL: recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. CoRR, abs/2004.03125.

4. SIGMOD '20: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data June 2020 Pages 2629–2636. https://doi.org/10.1145/3318464.3383128

5. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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.

6. Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning.

7. Jichuan Zeng, Xi Victoria Lin, Steven C.H. Hoi, Richard Socher, Caiming Xiong, Michael Lyu, Irwin King 2020. Photon: A Robust Cross-Domain Text-to-SQL System . In Proceedings of

the 58th Annual Meeting of the Association for Computational Linguistics, pages 204–214, Online. Association for Computational Linguistics.

8. Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio*:* Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

9. TensorFlow Tutorial : https://www.tensorflow.org/text/tutorials/nmt_with_attention