Auto-Query - A simple natural language to SQL query generator for an e-learning platform

Parth Parikh
Dept. of Computer Science & Engineering
Indian Institute of Technology Bombay
Mumbai, India
parthparikh28@outlook.com

Oishik Chatterjee
Dept. of Computer Science & Engineering
Indian Institute of Technology Bombay
Mumbai, India
oishik75@gmail.com

Muskan Jain
Computer Science & Engineering
Bharati Vidyapeeth's College of Engineering
New Delhi, India
anumi1999@gmail.com

Aman Harsh
Dept. of Mathematics & Computing
Indian Institute Of Technology Dhanhbad
Mumbai, India
aman.harsh.1451@gmail.com

Gaurav Shahani
Dept. of Computer Science & Engineering
Indian Institute of Technology Bombay
Mumbai, India
gaurav@e-yantra.org

Rathin Biswas
Dept. of Computer Science & Engineering
Indian Institute of Technology Bombay
Mumbai, India
r.biswas@iitb.ac.in

Kavi Arya
Dept. of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai, India
kavi@cse.iitb.ac.in

2022 IEEE Global Engineering Education Conference (EDUCON) | 978-1-6654-4434-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/EDUCON52537.2022.9766617

***Abstract*—Despite its difficulties, SQL is an essential tool for the users in an educational organisation who need quick and easy access to data to gauge the reception of their learning content by their students and potentially improve their content depending on the insights. To get these insights, the course instructors need real-time access to the database and also need to have relevant SQL knowledge to operate the database to retrieve the required data. The study explores ways to mitigate the difficulties of SQL by developing an application that takes natural language questions that the course instructors have and convert them into SQL queries using a sequence-to-sequence model that show them the data they asked for on a dashboard. The study found that there was a drastic reduction in the time it took for the users of the e-learning platform to get the data from the database without waiting for support from the database administrators. This in turn empowered the educators to study the data and get insights into the reception and working of the course and make suitable changes if necessary which might enhance the user experience for their students.**

***Index Terms*—SQL, Query-less data, SQL in Education, No syntax SQL, easy SQL**

We thank Ministry of Education (MoE), Govt. of India for sponsoring e-Yantra Project under National Mission on Education through Information and Communication Technology (NMEICT).

I. INTRODUCTION

Having the correct data enables educators to ask the right questions to their students and effectively curate their courses according to the standards and trends that the students set. It also gives the course instructors deep insights into the reception to their learning content and a potential to improve their learning materials and scaffold the learning [1] [2]. Because data is everywhere, it must be scrutinised and analysed objectively in order to transmit correct information. Collecting data in real-time and evaluating it facilitates differentiation and interventions wherever required [3]. All this data has to be stored somewhere [4]. To manage and organise data, most educational institutions employ spreadsheets and excel tools. However, as their use cases grow, they develop specialized in-house database systems [5]. The database management system has many benefits to the education sector [6]. Using a DBMS ensures that the data stored by institutions is safe and protected from information leakage and unauthorized access. It also improves efficiency during data migration and aids the institution to meet compliance. Due to the covid-19 pandemic, almost all institutions have developed their management systems [7]. With this dependence on databases to store data, it has become equally important for people working with the data to know the database management languages such as SQL. SQL is the de-facto standard programming language that manages the data and manipulates it from relational databases [8]. Even 47 years after it's inception, SQL remains one of the most popular languages to work with and was voted third on the 2020 Stackoverflow's developer survey [9]. Though SQL remains one of the most popular languages to work with while using a relational database [10], the language proves to be challenging to adapt to for non-programmers and even budding developers [11]. This is due to the fact that the language is strongly typed, and queries tend to get complex when retrieving and understanding large amounts of data. For a user to fetch any kind of meaningful data from multiple tables and showing the data in a helpful format requires complex queries, sometimes nested queries, and often result in errors that cannot be handled by anyone except experienced database administrators. There is an utter need to make the data retrieval process from databases easier in such a case.

II. Purpose of this Study

The study explores ways to mitigate these difficulties of SQL by developing an application that takes natural language questions, converts them into SQL queries and then shows the results on the e-learning dashboard using a sequence-to-sequence model [12] [13]. Whereas a natural language interface to SQL is the Holy Grail, many applications don't need such accuracy in a generic manner [14]. Consider an e-learning context where 90% of the queries to our data might be clustered into a small set of queries for which we could focus our training data set. It is this insight that has motivated this work.

III. Related Work

Machine learning and natural language processing emerged as an alternative to make this task easier. Text to SQL has emerged as a mechanism to convert the natural language statements into SQL queries. Since a SQL query statement is a sequence that conforms to the grammar and has a logical structure, we can divide it into three parts: the database, the question, and the SQL keyword, and most of the studies on this problem statement to date is focused on these three components of the SQL. In the current deep learning research context, the Text-to-SQL task can be regarded as a sequence-to-sequence generation task similar to neural machine translation, mainly using the Seq2Seq model framework. The initial researches showed that the typical Seq2Seq model could not perform better due to the difference in the encoding and decoding steps of the Seq2Seq model. In the encoding stage, a good alignment or mapping relationship needs to be formed between the question and the database, i.e., which elements in which tables are involved in the question (including column names and table element values). At the same time, the question and SQL syntax also need to be mapped. That is, which keyword operations (such as Group, Order, Select, Where, etc.) and aggregation operations (such as Min, Max, Count, etc.) are triggered by the words in the question; finally, the logical structure of the question expression needs to be expressed and fed back to the generated SQL. In the query statement, the logical structure includes nesting, multiple clauses, etc.

A. Practice Datasets

Several semantic parsing data sets with different queries have been created. Some of the dataset includes ATIS, GeoQuery , JOBS. Some other datasets that are present of text-to-SQL datasets also include Restaurants, Scholars, Acedamic, Yelp and IMDB, Advising and WikiSQL.

One of the standard datasets used in Text-to-SQL task is WikiSQL [15]. WikiSQL is the collection of the questions and corresponding SQL queries. The dataset contains corpus of 80654 SQL queries over 24241 schema. It's a supervised text-to-SQL dataset which Amazon Mechanical Turk hand-annotated flawlessly.

Another dataset that is in literature for the text to SQL problem is spider. Spider [16] dataset provides with a large set of complex and cross domain and complex SQL queries dataset. It consists of 200 databases with multiple tables, 10,891 questions and 5,691 corresponding SQL queries. The striking feature of this dataset is that it has classified its queries into easy, medium, hard and extra-hard queries.

B. Text to SQL

Text-to-SQL is a technique to automatically translate a user's query, spoken in natural language, into SQL. It falls under the semantic parsing — the task of converting natural language to a logical form. The first works were done on WikiSQL benchmark through Seq2SQL [15]. Seq2SQL is a deep neural network that converts natural language questions about a database into SQL queries. Seq2SQL takes advantage of SQL queries' intrinsic structure by encoding the question and table schema first, and then predicting each section of the query separately. A recurrent network is used to encode the question. Likewise, each column, which may consist of many words, is encoded via another recurrent network. The question encoding is also used along with the column encoding to compute the column for the selection operation. The where clause is generated using the augmented pointer network.

SQLNet [17], is to fundamentally solve "order-matters" problem by discarding the sequence-to-sequence structure when the order does not matter. It utilizes a sketch-based technique, in which the sketch consists of a dependency graph so that one prediction can be done by taking into consideration only the previous predictions on which it is dependent. It also proposes a sequence-to-set model and a column attention technique for synthesising queries based on the sketch. By combining all these novel strategies, SQLNet can outperform the prior state of the art by 9% to 13% on the WikiSQL task.

EditSQL [18] tries to tackle a context-dependent text to SQL query generation problem by using interaction histories and an utterance-table encoder-decoder unit that can reliably understand the context of a user's utterance (or question). To do so, they employ a BERT-based encoder capable of capturing complicated database structures and linking them to utterances.

As a result, given any query, the model will almost always properly identify the database schema to which the question matches.

RAT-SQL [19] is based on the relation-aware self-attention mechanism which enables address schema encoding, feature representation and schema linking within a text2SQL encoder. The main focus of the paper is on generalization problem of the Text to SQL model on an unseen database. For modeling text-to-SQL generation, the approach adopted the encoder-decoder framework. It provides a way to combine predefined hard schema relations and inferred soft self-attended relations in the same encoder architecture. The input is a graph in which each node is column and databases.

Semi-autoregressive Bottom-up Parser(SmBoP) [20] is a bottom-up parsing algorithm that builds the top-K sub-trees of height $<$ T at decoding step T. The parser constructs the top-K program sub-trees of depth $<$ T in parallel at each decoding step T (similar to beam search). This results in runtime complexity that is logarithmic in tree size rather than linear, contributing to the surge in interest in more efficient AI systems. Given a beam Z_t with K = 4 trees of height t (blue vectors), we make use of cross-attention to contextualize the trees with information from the input question. We then score the frontier which is the set of all trees of height t+1 that can be constructed using grammar from the current beam, and the top-K trees are kept. Finally, for each of the new K trees, a representation is created and placed in the new beam $Z_t + 1$. The parser returns the highest-scoring tree after T decoding steps in Z_t that corresponds to a full program. Because we have gold trees at training time, the entire model is trained jointly using maximum likelihood.

IV. Methodology

The study uses a dataset from e-Yantra Robotics Competition (eYRC), a popular national level competition hosted by the Indian Institute of Technology Bombay to retrieve data using RAT SQL and Semi-autoregressive Bottom-up Parser(SmBoP) as base models. Firstly we created a dataset by containing natural language questions and the corresponding sql queries on the eYRC databases. After that we trained the SmBoP model (since it was the state-of-the-art) on two datasets:

1) eYRC + spider
2) eYRC

In the following sub-sections we describe the data creation process in details.

A. Dataset Preparation

The dataset consists of two parts - i) database and ii) questions and SQL queries. Firstly we converted the eYRC database into similar format of the spider database. For the same, we first converted the database into spider table format. We then set up the foreign key relations and primary key for the internal database.

B. Question and SQL Annotations

We collected some of the natural language questions that were demanded to be solved with our project. These questions were converted into different ways to add diversity of asking question. We didn't employ any template or script to generate questions and SQL queries since we wanted them to be diverse, realistic, and reflective of how people actually use databases.

Now each question was batched into same group of asking question. A SQL query was created of each batch and was verified by running the query on the database and matched with the expected results. We make certain that our corpus contains sufficient samples of all frequent SQL patterns that may be seen in the future while dealing with the database.

C. Model and Experimental Setup

As previously stated, we have used the SmBoP architecture for training the Text-to-SQL task. We use a RAT-SQL encoder to encode the input and schema, which consists of 24 Transformer layers followed by 8 RAT-SQL layers. The decoder is SmBoP. Our pre-trained model had the beam size of K = 30, and the number of decoding steps are T = 9 at inference time, which is the maximal tree depth on the development set. The tree representation transformer has one layer, eight heads, and a dimensionality of 256. We train with a batch size of 8 and halt training early based on the development set. The fine tuning of the model was done on our dataset.

We loaded the initial weights of the model that were best on the spider dataset and preprocessed our data and provided the model with that data. The process continued for the spider + eYRC dataset and then only for the eYRC dataset. To boost accuracy, we experimented with the learning rate of the dataset and the batch size and varied beam size from 25 to 30 to look for improvements in accuracy.

V. Results

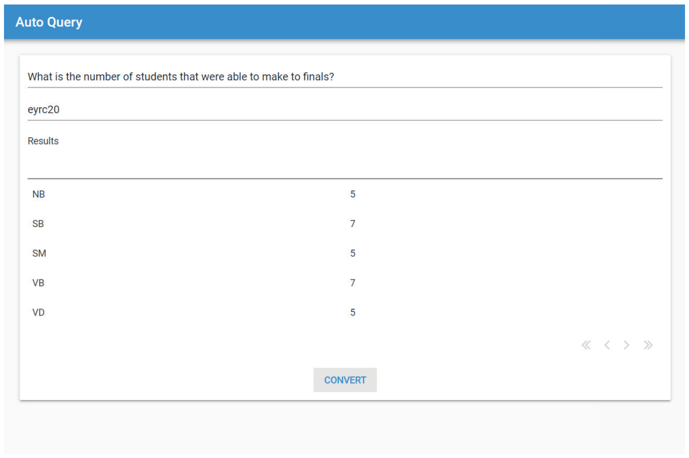
The natural language questions to be solved are collected from the e-Yantra's office staff who had little to no knowledge using SQL. After different experimentation on different datasets and different parameters, following results were obtained. An accuracy of 69.8% on the combined spider and eYRC dataset was observed. The study further noticed an accuracy of 68.75% when trained on the eYRC dataset only.

VI. Discussion

The study investigates the different methods and tools that can be used to make data retrieval faster and efficient without needing database developers to write queries to get data. After a detailed analysis and testing done by the office staff at e-Yantra, the experience they shared strengthen the need for this study. The test group reported that it was much more easier to get simple data in the form of tables and graphs just by asking the system to represent the required data, in simple natural language sentences. There was also a drastic change in the time it took for them to get the data and study it. Normally, if the office staff required any data, they would have

Question	Expected Queries	Generated Queries
Number of registrations from college id DL1005	SELECT COUNT(DISTINCT id) FROM team_details WHERE college_id = 'DL1005'	SELECT COUNT(*) FROM team_details WHERE team_details.college id = 'DL1005'
How many teams have registered in eYRC till date?	SELECT COUNT(DISTINCT id) FROM team_details	SELECT COUNT(DISTINCT team_details.id) FROM team_details
Tell me the colleges from Karnataka	SELECT college_name FROM college_list WHERE state= 'Karnataka';	SELECT college_list.college_name FROM college_list WHERE college_list.state = 'Karnataka'
What are the college ids from Delhi?	SELECT clg_code FROM college_list WHERE state = 'Delhi';	SELECT college_list.clg_code FROM college_list WHERE college_list.state = 'Delhi'
The college id of ABCITS is?	SELECT clg_code FROM college_list WHERE college_name='ABCITS';	SELECT college_list.clg code FROM college_list WHERE college_list.college_name = 'ABCITS'
How many teams have registered in eYRC from KL1024, DL1030, UP1096, AP1050?	SELECT COUNT(id) FROM team details WHERE college_id = 'KL1024' OR college id = 'DL1030' OR college_id = 'UP1096' OR college_id = 'AP1050';	SELECT COUNT(DISTINCT team_details.id) FROM team_details WHERE team_details.college_id = 'KL1024' OR team_details.college_id = 'AP1050'

TABLE I
SOME QUESTIONS ASKED FROM THE MODEL ALONG WITH THE EXPECTED QUERIES FROM MODEL AND GENERATED QUERIES OF MODEL


Auto Query

What is the number of students that were able to make to finals?

eyrc20

Results

```
NB    5
SB    7
SM    5
VB    7
VD    5
```

Fig. 1. A natural language question generating the query

to depend on the developers with the technical know-hows of the database design and ask them to retrieve the data for them which they would then visualise in spreadsheets and data processing applications. The application that emerged from this study essentially eliminates the need of the middle man while getting data from the database. Thus it has improved their productivity as well as make a stark decrease in the time required to study the latest trends of their e-learning courses and take appropriate interventions to improve the quality of experience for their courses' students.

VII. Limitations

The results presented in this study are based on use case of the office staff and an e-learning company, the feedback that was taken from the staff at the end of one of their courses and the analysis was presented based on the usability findings of that course. This limits the generalization of findings to this scope and the pedagogy in other courses are required to be explored in order to generalise the findings across all domains and work towards improving the accuracy [21].

References

[1] B. Williamson, "Learning in the 'platform society': Disassembling an educational data assemblage," *Research in Education*, vol. 98, no. 1, pp. 59–82, 2017.
[2] I. Nishane, V. Sabanwar, T. G. Lakshmi, D. Singh, and R. Rajendran, "Learning about learners: Understanding learner behaviours in software conceptual design tele," in *2021 International Conference on Advanced Learning Technologies (ICALT)*, 2021, pp. 297–301.
[3] K. Vassakis, E. Petrakis, and I. Kopanakis, "Big data analytics: applications, prospects and challenges," in *Mobile big data*. Springer, 2018, pp. 3–20.
[4] H. J. Watson, "Tutorial: Big data analytics: Concepts, technologies, and applications," *Communications of the Association for Information Systems*, vol. 34, no. 1, p. 65, 2014.
[5] H. West and G. Green, "Because excel will mind me! the state of constituent data management in small nonprofit organizations," 2008.
[6] S. Dahiya, S. Jaggi, K. Chaturvedi, A. Bhardwaj, R. Goyal, and C. Varghese, "An elearning system for agricultural education," *Indian Research Journal of Extension Education*, vol. 12, no. 3, pp. 132–135, 2016.
[7] L. Mishra, T. Gupta, and A. Shree, "Online teaching-learning in higher education during lockdown period of covid-19 pandemic," *International Journal of Educational Research Open*, vol. 1, p. 100012, 2020.

[8] R. Batra, "A history of sql and relational databases," in *SQL Primer*. Springer, 2018, pp. 183–187.
[9] S. L. Vadlamani and O. Baysal, "Studying software developer expertise and contributions in stack overflow and github," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020, pp. 312–323.
[10] K. Affolter, K. Stockinger, and A. Bernstein, "A comparative survey of recent natural language interfaces for databases," *The VLDB Journal*, vol. 28, no. 5, pp. 793–819, 2019.
[11] P. Rao and S. K. Kopparapu, *Friendly Interfaces Between Humans and Machines*. Springer, 2018.
[12] G. Katsogiannis-Meimarakis and G. Koutrika, "A deep dive into deep learning approaches for text-to-sql systems," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2846–2851.
[13] D. Shah, A. Das, A. Shahane, D. Parikh, and P. Bari, "Speakql natural language to sql," in *ITM Web of Conferences*, vol. 40. EDP Sciences, 2021, p. 03018.
[14] O. Gkini, T. Belmpas, G. Koutrika, and Y. Ioannidis, "An in-depth benchmarking of text-to-sql systems," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 632–644.
[15] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.
[16] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," *arXiv preprint arXiv:1809.08887*, 2018.
[17] X. Xu, C. Liu, and D. Song, "Sqlnet: Generating structured queries from natural language without reinforcement learning," *arXiv preprint arXiv:1711.04436*, 2017.
[18] B. Chandra, A. Banerjee, U. Hazra, M. Joseph, and S. Sudarshan, "Edit based grading of sql queries," in *8th ACM IKDD CODS and 26th COMAD*, 2021, pp. 56–64.
[19] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, "Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers," *arXiv preprint arXiv:1911.04942*, 2019.
[20] O. Rubin and J. Berant, "Smbop: Semi-autoregressive bottom-up semantic parsing," *arXiv preprint arXiv:2010.12412*, 2020.
[21] B. Hui, X. Shi, R. Geng, B. Li, Y. Li, J. Sun, and X. Zhu, "Improving text-to-sql with schema dependency learning," *arXiv preprint arXiv:2103.04399*, 2021.