IJIST
P ISSN : 2618-1630
E ISSN : 2909-6130

Natural Language to SQL Queries: A Review

Muhammad Shahzaib Baig[1], Azhar Imran[1], Aman Ullah Yasin[1], Abdul Haleem Butt[1], Muhammad Imran Khan[1]
[1]Department of Creative Technologies, Faculty of Computing & AI, Air University, Islamabad
*Correspondence: Dr. Azhar Imran, azharimran63@gmail.com

Citation | Baig. M. S, Imran. A, Yasin. A. U, Butt. A. H, Khan. M. I, "*Natural Language to SQL Queries: A Review*". International Journal of Innovations in Science and Technology, Vol 4, Issue 1, pp: 147-162, 2022.
Received | Jan 3, 2022; Revised | Feb 10, 2022 Accepted | Feb 12, 2022; Published | Feb 22, 2022.

Abstract.
The relational database is the way of maintaining, storing, and accessing structured data but in order to access the data in that database the queries need to be translated in the format of SQL queries. Using natural language rather than SQL has introduced the advancement of a new kind of handling strategy called Natural Language Interface to Database frameworks (NLIDB). NLIDB is a stage towards the turn of events of clever data set frameworks (IDBS) to upgrade the clients in performing adaptable questioning in data sets. A model that can deduce relational database queries from natural language. Advanced neural algorithms synthesize the end-to-end SQL to text relation which results in the accuracy of 80% on the publicly available datasets. In this paper, we reviewed the existing framework and compared them based on the aggregation classifier, select column pointer, and the clause pointer. Furthermore, we discussed the role of semantic parsing and neural algorithm's contribution in predicting the aggregation, column pointer, and clause pointer. In particular, people with limited background knowledge are unable to access databases with ease. Using natural language interfaces for relational databases is the solution to make natural language to SQL queries. This paper presents a review of the existing framework to process natural language to SQL queries and we will also cover some of the speech to SQL model in discussion section, in order to understand their framework and to highlight the limitations in the existing models.

Keywords: Natural Language Processing, Structured Query Language (SQL), Text to Relational database, Natural Language Interface for Databases (NLIDB), Intelligent Database System (IDBS), Flexible Querying, Database Management System (DBMS).

INTRODUCTION

Natural language interfacing to a relational database is a challenging and important problem to a coupe. It requires a model that can understand natural language and can translate it into relational database queries structures. Using natural language rather than SQL has introduced the advancement of a new kind of handling strategy called Natural Language Interface to Database frameworks (NLIDB). NLIDB is a stage towards the turn of events of clever data set frameworks (IDBS) to upgrade the clients in performing adaptable questioning in data sets [9]. A model that can deduce relational database queries from natural language. Advanced neural algorithms synthesize the end-to-end SQL to text relation which results in the accuracy of 80% on the publicly available datasets [1].

In this paper, we will be covering the existing framework and compared based on the aggregation classifier, select column pointer, and the clause pointers and we will be looking into the role of semantic parsing and neural algorithm's contribution in predicting the aggregation, column pointer, and clause pointer that is the existing solution to the problem. The LUNAR and LADDER systems were intended for non-technical users to pose natural language queries regarding moon rock samples and US Navy ships, respectively, in the 1970s (Woods, 1972) [2]. The fast advancement of computer hardware and software over the last five decades has had such an impact on databases that database systems established in the 1970s are no longer even compatible with the current definition of a database (Bercich 2003; Frank 2018). To meet the industry's demands, various natural language to database querying frameworks have been created since then. We discovered intriguing research trends in converting natural language to database queries domain retrieval time, reduced support for language portability, and difficult configuration processes by evaluating the development timeline of such systems [2].

It is not an easy process to translate a natural language question into multiple database query languages like SQL, Simple Protocol, and RDF Query Language (SPARQL), because today's databases are diversified, massive in size, and use complex data storing technologies (Nadkarni, 2011) [3]. Storage engines frequently store data in several formats, including structured (tabular), No SQL or graph (text), and hybrid. As a result, distinct query languages are required by underlying storage engines to obtain the stored data [4]. The diversity of data storage technologies complicates natural language to database query translation. With the evolution of machine learning techniques, many frameworks that can efficiently interpret natural language inquiries have been developed.

Recent work shows that recurrent neural networks stacked with attention and copying mechanism have greater potential in semantic parsing. The Seq2SQL model shows that utilizing separate decoders for various pieces of an inquiry (i.e., accumulation activity, target section, and where predicates) expands forecast precision, and supports learning further works on the model by permitting it to adapt semantically comparable inquiries past management.

In current study we have categorized the existing frameworks into subcategories based on their implementation techniques for comparing their results. Frameworks are categorized and discussed as SQL-Net, Syntax SQL-Net, Grammar SQL, IR-Net, Edit-SQL, and RAT-SQL based on their algorithms and performance.

After evaluating the existing models for natural language to SQL, we have discussed existing solutions to perform the speech to SQL queries and overviewed their model in short and we will also discuss the Debug-It-Yourself (DIY) model in the analysis and discussion section to evaluate their limitation and advantages along with hurdles to use them as a practically deploying model for real case scenario, followed up after all this the we will look into the future extension of the work and improvements in the existing problem in view of author as closing remark of the discussing section which includes ontological acyclic directed graphs as one of the solution to improve the accuracy of such systems.

Existing Frameworks

The structured query language is the most used language for retrieving data from the structured database, various methods are included in the no-SQL category that are rule-based syntax analysis, semantic analysis, and pattern matching. Figure 1 shows the conventional tree for natural language to database approach and Figure 2 discusses the idea of seq2sql that is generating SQL queries using reinforced learning [4]. Furthermore, we discussed the existing framework for the problem as SQL-Net, Syntax SQL-Net, Grammar SQL, IR-Net, Edit-SQL, and RAT-SQL.

SQL-Net

The SQL-Net model was introduced to minimize the use of the reinforced learning as shown in Figure 2, by avoiding seq2seq in cases where an order is of low significance. SQL-Net is based on a sketch-based approach which is comprised of the dependency graph based on previous predictions so that previous predictions can be used in the account of predicting the next queries reducing the use of reinforced learning [3]. WikiSQL, one of the states of art models uses the SQL-Net approach to solve this particular problem. Figure 3 shows the framework of WikiSQL. The SQL-Net model also used a weight-based approach to highlight and give more attention to the significant words and phrases in sentences. SQL-Net uses the concept of sequence to set prediction which predicts the name of the columns [5]. It is a part of the generic attention mechanism for feature attention map on question-based on column based. Predicting the "Where" clause is the most challenging part of the processing [6].

SQL-Net searches for the "Where" clause and each chosen column and constraint by predicting the operation and value using a sketch-based approach. Accuracy of SQL-Net on Wiki-SQL test set is 64.4% and on SPIDER test set is around 12.4%. [6]. Seq2SQL framework inefficiencies (generalized to unseen schema and serializability) and its Seq-to-Seq model have been improved with the aid of a new technique, namely Xiaojun's sequence-to-set based model (Xu, 2017). The model was implemented in the "SQL-NET" program, which uses Seq2SQL as a baseline foundation but does not use reinforcement learning. A sketch-based technique, similar to SQLizer, has been designed to parse the NL query, but each sketch contains a dependency neural network to predict the new sketch based on a prior prediction of a sketch. This new model increases Seq2SQL outcomes by 9% to 13% across many measures [7].

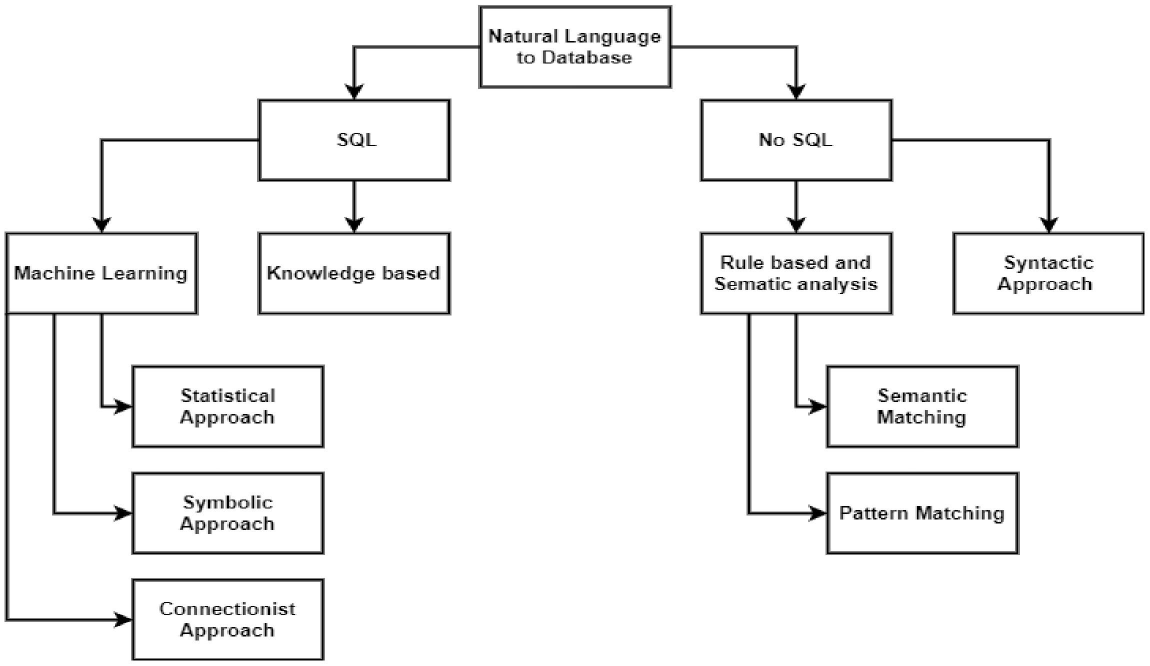

Figure 1. Classification of natural language to SQL query framework [7].

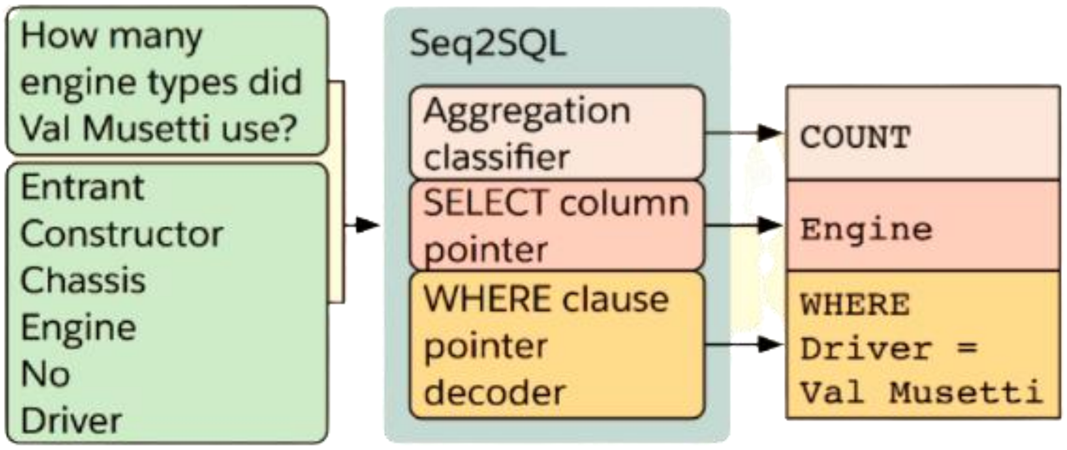

Figure 2. Frame of Natural Language to SQL queries using reinforced learning [5].

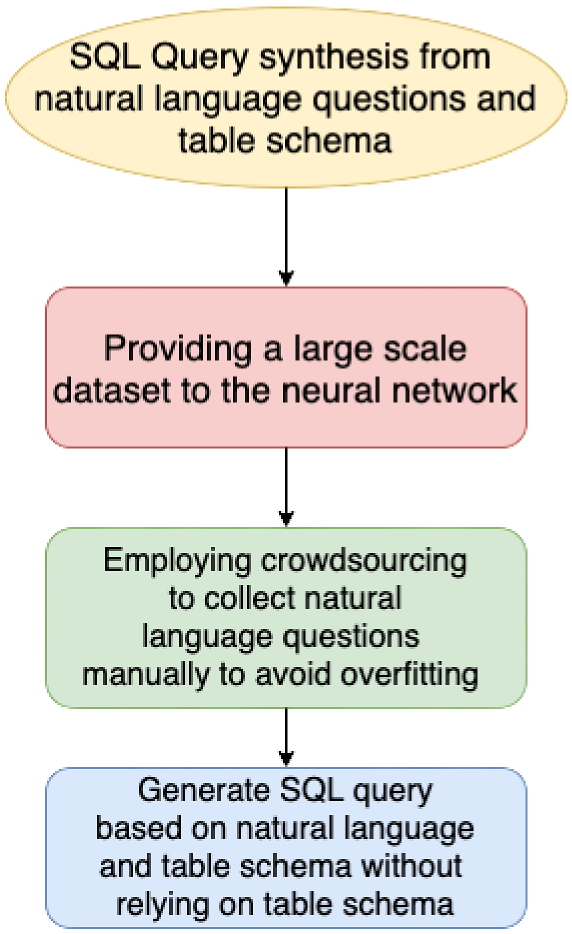
SQL Query synthesis from natural language questions and table schema
Providing a large scale dataset to the neural network
Employing crowdsourcing to collect natural language questions manually to avoid overfitting
Generate SQL query based on natural language and table schema without relying on table schema

Figure 3. Frame of WikiSQL using SQL-Net [10].

Syntax SQL-Net

Syntax SQL-Net was designed to generate complex queries with multiple clauses and cross-domain queries, generalizing previous models to complex database structures by using the syntax tree networks shown in Figure 4. It uses table-aware encoders and decoders having SQL generation path history. For handling complex queries at decoders, the syntax SQL-Net uses SQL grammar for determining modules to be invoked at each recursive step [9]. Syntax SQL-Net was assessed on the SPIDER dataset and surpassed the previous model in precision by 7.3% [3].

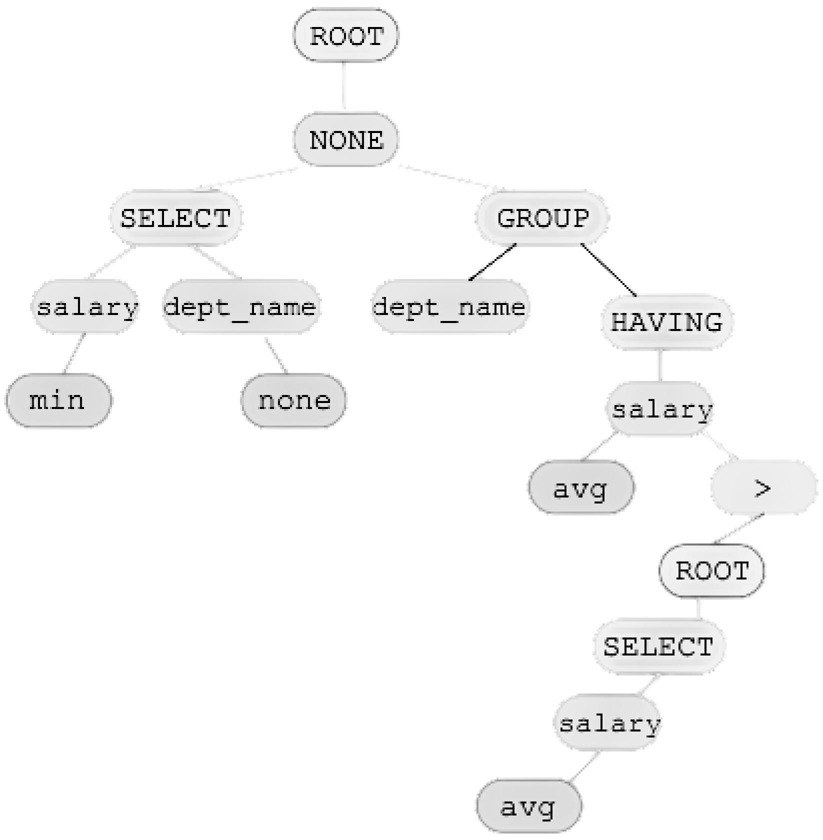

Figure 4. Tree-based query generator using syntax SQL-Net [10]

Grammar SQL

The sequence-to-sequence paradigm used by neural text-to-SQL models usually decodes at the token level and does not contemplate producing SQL hierarchically from a grammar. The model provides a schema-dependent language with minimal over-generation [20]. The shallow parsing expression language is intended to capture the least amount of SQL required to cover most of the cases in the dataset. Later advancements introduced non-terminals to context-free grammar for context-sensitive SQL components to maintain consistency of table, column, and value references [25]. The model utilizes runtime restrictions during decoding to ensure that only legitimate programmers are allowed to link separate tables together using a foreign key. As input, the suggested model accepts a natural language utterance, a database, and grammar relevant to that speech. The SQL query is then built up repeatedly by the decoder utilizing the attention technique on the input sequence [26]. Anonymization of identifiers is also used to replace database IDs that occur in both the natural language inquiry and the SQL query. Figure 5 shows the implementation of Grammar SQL [27].

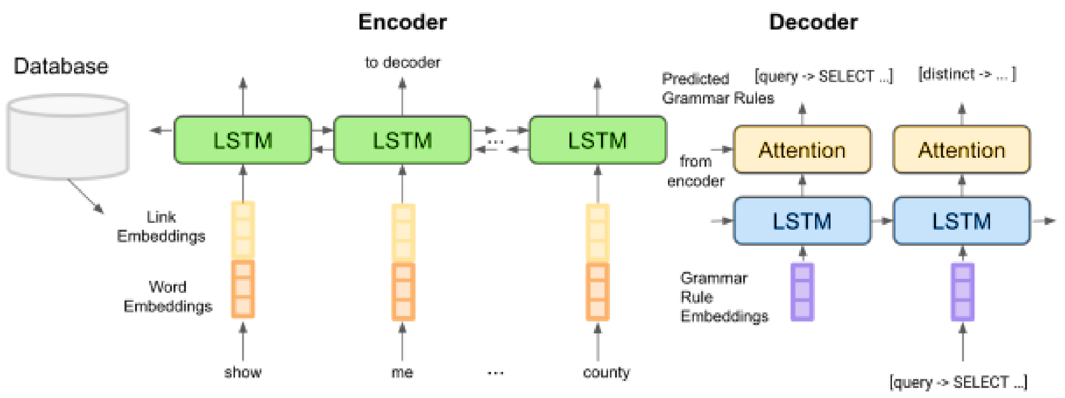

Figure 5. Grammar SQL proposed model [13]

IR-Net

This model uses a neural approach and called Intermediate Representation (IR) shown in Figure 6 [16] and it mainly focuses on two main problems in the existing models i.e. the mismatch between intents in text and the prediction of column names caused by a tremendous number of out of domain words. Instead of synthesizing SQL queries end-to-end, IR-Net decomposes natural language into three parts. The schema linking procedure over a database schema and a question are carried out during the first phase [31]. IR-Net makes use of Sem-QL, a domain-specific language that acts as a bridge between SQL and plain language. Each model component performs a unique role in the Text2SQL job. Natural language input is encoded into an embedding vector using the NL encoder. These embedding vectors are used to build hidden states with a bi-directional LSTM. The schema encoder takes a database schema as input and produces column and table representations. Finally, context-free grammar is employed by the decoder to construct Sem-QL queries [36].

Zhong et al. introduced "Seq2SQL," a neural network-based framework for interpreting natural language queries and mapping them to SQL (Structured Query Language) representations (Zhong, Xiong, and Socher 2017). The suggested system minimizes query space and enhances system execution accuracy. This framework used Reinforcement Learning (RL) rewards and cross-entropy loss iteratively on query execution over the database to create unordered parts of the question that are less suitable for advancement using cross-entropy misfortune. They released WikiSQL, a dataset including 87673 hand-explained examples of inquiries and SQL questions spread across 26521 tables from Wikipedia. Seq2SQL outperforms the best in class semantic parser by Dong and Lapata by incorporating strategy-based reinforcement learning (RL) with an inquiry execution condition into WikiSQL (2016) [39].

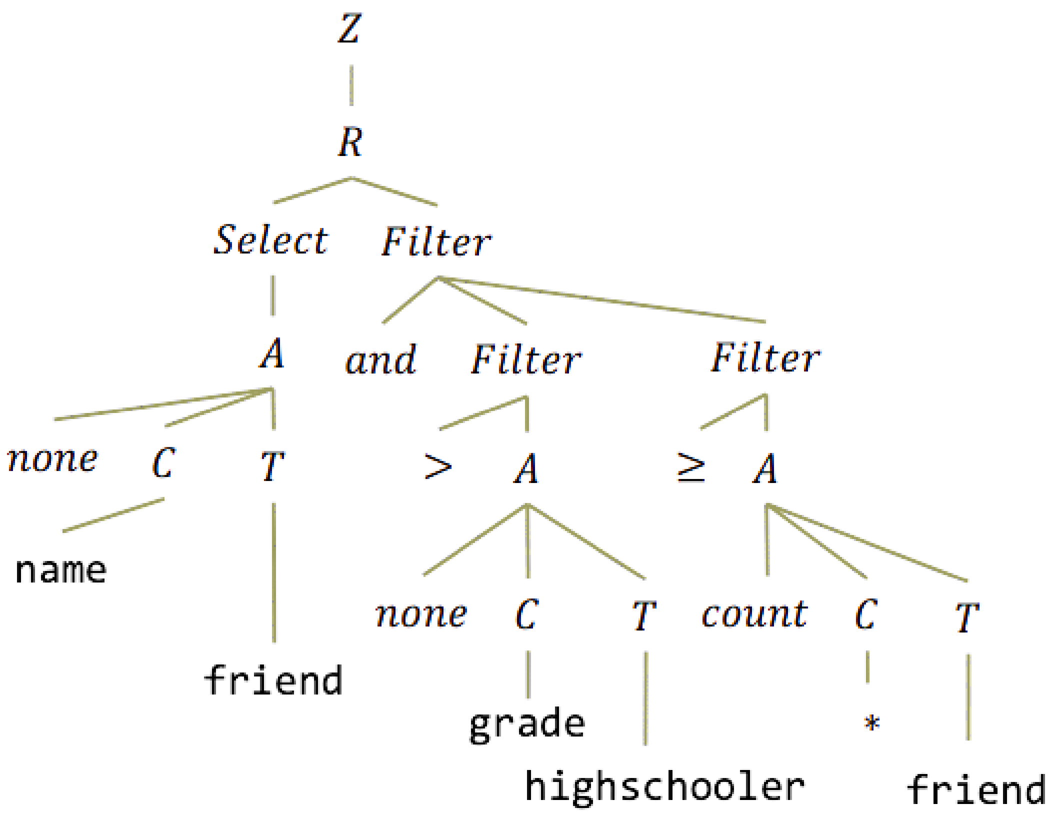

Figure 6. Graphical implementation of IR-Net

Edit-SQL

Edit-SQL [16] is primarily concerned with the cross-domain context-dependent text-to-SQL operation as shown in Figure 7. The authors take advantage of the fact that adjacent natural language questions are dependent on one another and that corresponding SQL queries overlap. They make use of this reliance by modifying the previously anticipated query in order to increase the generating quality. The editing mechanism accepts SQL sequences as input and reuses generating results at the token level. To cope with complicated tables in many domains, an utterance-table encoder, and a table-aware decoder is utilized to include the natural language context and the schema [34]. User utterances and table structure are encoded using the utterance-table encoder. To encode speech tokens, a bi-LSTM is employed.

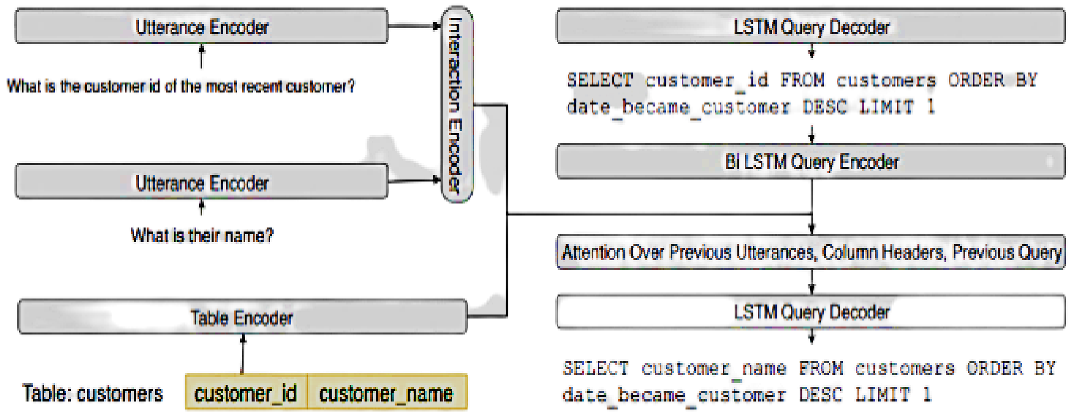
Utterance Encoder
What is the customer id of the most recent customer?
Utterance Encoder
What is their name?
Interaction Encoder
Table Encoder
Table: customers
customer_id
customer_name
LSTM Query Decoder
SELECT customer_id FROM customers ORDER BY date_became_customer DESC LIMIT 1
Bi LSTM Query Encoder
Attention Over Previous Utterances, Column Headers, Previous Query
LSTM Query Decoder
SELECT customer_name FROM customers ORDER BY date_became_customer DESC LIMIT 1

Figure 7. Architecture of Edit-SQL form [8]

RAT-SQL

One of the most difficult aspects of converting natural language inquiries into SQL queries is generalizing to unknown database schemas, flowchart for RAT-SQL is shown below in figure 8. The generalization is dependent on encoding the database relations in an accessible manner and modeling alignment between important database columns in the query. The suggested architecture is built on the relation-aware self-attention technique, which enables address schema encoding, feature representation, and schema linking inside a text2SQL encoder [7]. Examine the flow chart below to gain a fast idea of RAT-encoder-decoder SQL's structure.

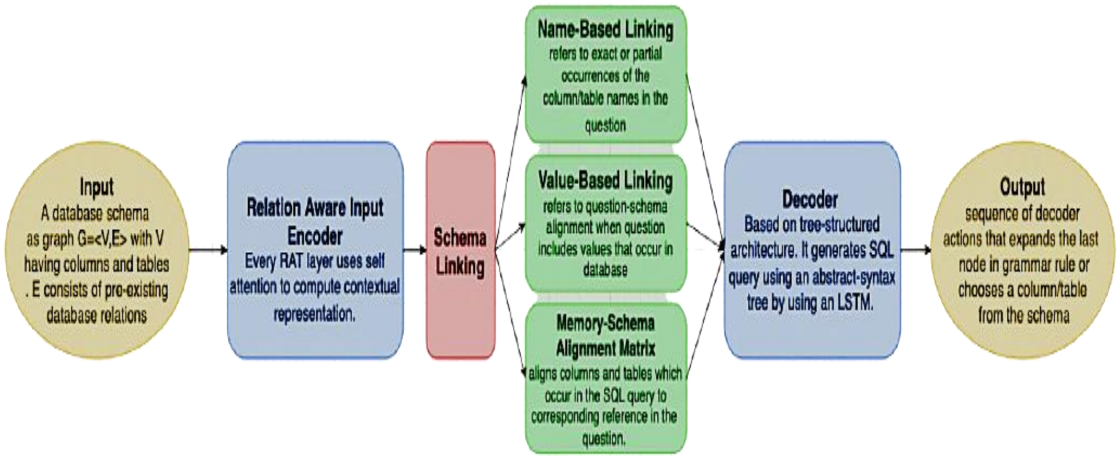
Input
A database schema as graph G=<V,E> with V having columns and tables . E consists of pre-existing database relations
Relation Aware Input Encoder
Every RAT layer uses self attention to compute contextual representation.
Schema Linking
Name-Based Linking
refers to exact or partial occurrences of the column/table names in the question
Value-Based Linking
refers to question-schema alignment when question includes values that occur in database
Memory-Schema Alignment Matrix
aligns columns and tables which occur in the SQL query to corresponding reference in the question.
Decoder
Based on tree-structured architecture. It generates SQL query using an abstract-syntax tree by using an LSTM.
Output
sequence of decoder actions that expands the last node in grammar rule or chooses a column/table from the schema

Figure 8. Flowchart of RAT-SQL [7]

Analysis and Discussion

The models discussed above have their logics and algorithms to be considered. The main objective of all the models is to improve the accuracy of the natural language to SQL queries. We have compared all the mentioned models on the SPIDER dataset. Spider is a large-scale, complicated, cross-domain semantic parsing and text-to-SQL dataset. It comprises of 10,181 questions and 5,693 distinct sophisticated SQL queries on 200 databases with numerous tables across 138 domains. In Spider, train and test sets contain a variety of complicated SQL queries and databases. To perform effectively on it, systems must be able to generalize not only to new SQL queries but also to new database structures. SQL-Net models are designed using sketch-based algorithms and reducing the use of reinforced learning to handle simple queries without focusing on cross-domain SQL queries. Dar et al [2] calculated the accuracy of SQL-Net on different datasets and the accuracy of SQL-Net on Wiki-SQL test set is 64.4% and on SPIDER test set is around 12.4%. Syntax SQL-Net surpasses the SQL-Net in the accuracy of 15% absolute increase in the accuracy compared to SQL-Net on SPIDER dataset, the accuracy of the SQL-Net is 12.4% while the accuracy observed by Tao Yu et al [10] is 27.2% in SPIDER dataset by also handling with complex and cross-domain queries.

Grammar SQL uses long short-term memory to handle natural language to SQL queries. Grammar SQL can handle complex and cross-domain queries along with good handling of denotation up to an accuracy of 80% and total accuracy calculated by Kevin Lin et al is 52% on the SPIDER dataset. According to Jiaqi Guo et al., IRNet surpasses all baselines by a significant margin. On the test set, it outperforms Syntax SQLNet by 27.0 percent. It also outperforms Syntax SQLNet, which conducts large-scale data augmentation, by 19.5 percent. When BERT is used, the performance of both Syntax SQLNet and IRNet improves significantly, and the accuracy gap between them on both the development and test sets widens and the total accuracy is 44.5% but Jiaqi Guo et al also used the IRNet with BERT model which gives the accuracy of 52.325%.

According to Zhang et al, the Edit-SQL model in cross-domain text2SQL generation achieves 32.9% accuracy. Furthermore, using BERT embedding results in a considerable gain in accuracy, attaining 53.4%. Wang et al mention RAT-SQL outperforms prior benchmark models by 8.7% and achieves 57.2% on the SPIDER dataset. Incorporating RAT-SQL with BERT results in a significant boost in performance and 65.6% accuracy.

The model discussed above have their pros and cons in the context of application, accuracy and the computational complexity. Models discussed above as their measure of accuracy increases, they would seem to have more computational complexity by using ensembling techniques [36]. As discussed earlier, the highest performing model for under discussion task is the RAT-SQL ensembled with BERT model achieving the accuracy of 65.6% on cross-domain SQL queries but RAT-SQL uses the linking schema with relation-based encoder and decoder with BERT increases its computation complexity [27]. While discussing in context of computational complexity is less for syntax SQL-Net as it used syntax-based decision trees as compared to the others but it only achieves the overall accuracy of 12.4% [28]. If we consider the computational complexity along with the performance the IR-Net with BERT model seem to perform good with over 50% accuracy but lesser computational complexity [29].

Table 1. Comparison of accuracies of all frameworks

Dataset	Model	Overall Accuracy
SPIDER	SQL-Net	12.4%
SPIDER	Syntax SQL-Net	27.2%
SPIDER	Grammar SQL	52%

SPIDER	IR-Net	44.5%
SPIDER	IR-Net with BERT	52.35%
SPIDER	Edit-SQL	32.9%
SPIDER	Edit-SQL with BERT	53.4%
SPIDER	RAT-SQL	57.2%
SPIDER	RAT-SQL with BERT	65.6%

While looking at the extensions in the existed model, Arpit Narechania et al [49] discussed the important of importance of natural language to structured queries mentioned the Debug-It-Yourself (DIY) approach based on state of art model NL2SQL in which they provided user with a sandbox where user have the provision to interact with three elements those are the mapping of the generated query on the basis of given question, small subset of underlying database and ensembled modal explanation of generated query, their discussion is the user interaction with the sandbox seems to efficient for control environment testing but not for actually deploying the models in professional environment where the feasibility and accessibility come into play. My analysis for DIY approach using NL2SQL is strongly based on the argument that the whole point of using systems like NL2SQL is to improve the feasibility element for the individual who do not have expertise on SQL or any structured query, assumingly user interaction in DIY is at that state which user is not familiar with either so Debug-It-Yourself is not a part of benchmark evaluation. Serval authors like Blanning et al [47] discussed this problem in context of semantic parsing in natural language processing using LSTM or Bi-LSTM models but the issue that persists is the complex queries or cross-domain queries, considering it into account it performs sufficiently good but it cannot handle cross-domain SQL queries.

Another method discussed by Song et al [47] comprised on the speech to SQL conversion in which they discussed that voice communication is far better approach as compared to the user giving (typed inputs) to the system and then system converts the input to SQL query. They mentioned that voice-based assistants for such purposes are usually 6.7 times faster than the conventional input method. They mentioned voice-based assistants like Google Home, Siri and Cortana have emerged the market leading to much more feasible and accessible human computer interactions for general purposes while considered the under-discussion problem, Song et al., mentioned some existing models for speech to SQL query conversion, in which they mentioned SpeakQL model which provides the speech to SQL conversion but this model restricts the voice to be user to be as query which in my opinion is not an natural language to SQL but instead SQL query in voice to text deploying it as speech to text.

The other model Song et al., mentioned is EchoQuery which is purposed for same purpose that is speech to SQL conversion but EchoQuery follows the pattern of query as "What is the {Aggregation}{Columns(s)} of {Table(s)}?", and SpeakQL requires the query to be an exact SQL statement such as "Select Salary From Employees Where Name Equals John". So far the models discussed in speech is also not converting the free speech style to SQL queries so Song el al purposed a solution to the problem which is based on the Speech Encoder, Schema Encoder, Speech Schema Relation aware encoder and SQL aware decoder which they tested on benchmarks and gave a cumulative accuracy of 15.29% on speech to text conversion but what is the impact of accents, slangs and dialects on the results they have not either tested or at least mentioned it.

These models dealing with speech to SQL are discussed here to globalize the scope of the study as we are focused on the text to SQL query for this study. As a closing remarks in discussion section, as the results shows that right at the moment there is no such Natural Language Model exists which can handle or process the natural language to structured query efficiently, till now the reason I can see here is the ambiguity in the language or in other terms you can say it as that single sentence of English can have a lot of meaning depending on its context and many expressions which cumulatively defines the meaning and the structure of that particular sentence, so the context of the sentence may also lead to some false SQL query cases. For future work extension, the directive graphs or ontologies can be used to test the mentioned problems, it might reduce the problem as the directive graphs defines the relation, context and structure of sentence to an extent and I have not witnessed any research paper in this domain. An example of ontological directive graphs is shown in the Figure 9.

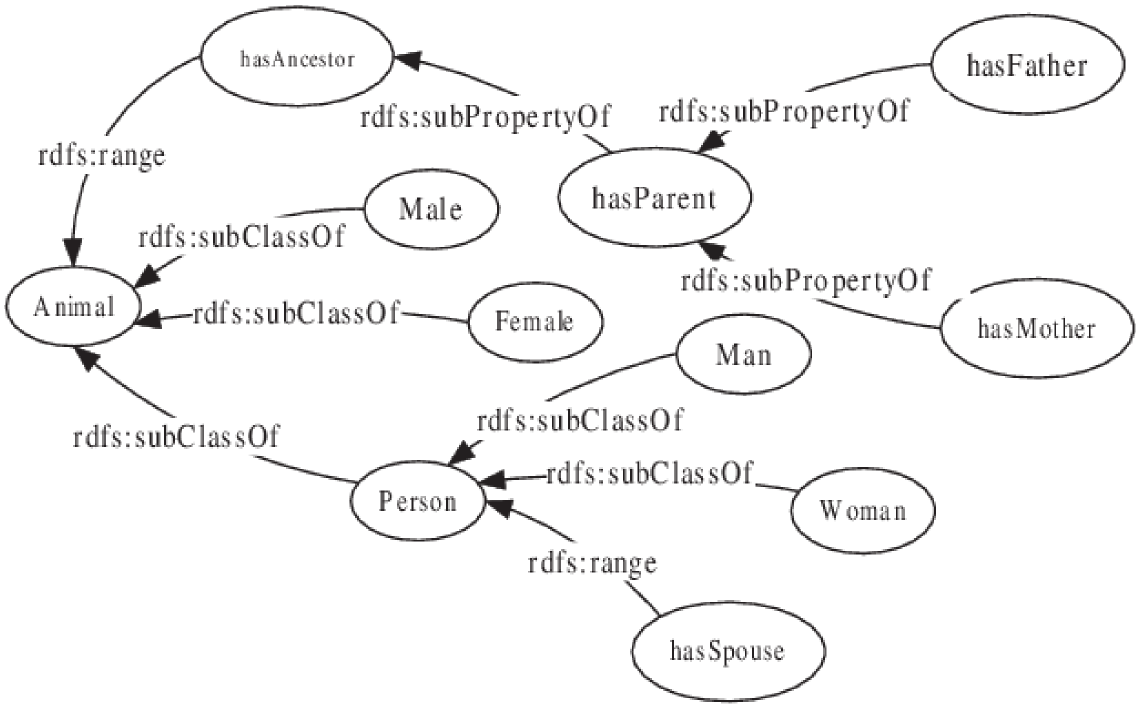

Figure 9. Ontological Directive Graphs [50]

Conclusion

In this paper, we reviewed the framework to process the natural language to simple and complex cross-domain queries. Furthermore, we have compared their algorithms and compared the accuracy of all the models on the same dataset that is SPIDER in our discussion, the dataset was kept the same throughout the discussion to evaluate all the frameworks on the same benchmark. The results mentioned in Table 1 show that RAT-SQL model with BERT model is outperforming all the pre-existing frameworks with an accuracy of 65.6% and Edit-SQL with BERT model standing at second place with an accuracy of 53.4%. In future the RAT-SQL model can be further improved to achieve higher accuracies in the context of aggregator and in the column prediction. We also discussed Debug-It-Yourself (DIY) model based on NL2SQL which improves user experience but have its own limitation relating to the prior experience of the user so DIY model is not specifically used for individuals with no prior or limited prior experience. We also discussed the speech to SQL query models in which the state of art model till now is proposed by Song el al, this speech to SQL query is not in the scope of the study but we have touched their limitations and advantages in context of accessibility and feasibility in analysis and discussion section, so far, the speech to text model seems to perform in limited and restriction environment. The future work we can extend on the basis of prior studies is to extend the work of natural language to SQL query using the ontological acyclic directed graphs for processing the sentences to identify and understand the contextual information.

Acknowledgment. The author expresses gratitude to the Department of Creative Technologies, Air University, Islamabad, under which the research has been carried out. Furthermore, all the results mentioned in this paper have been credited to their original authors and their work is mentioned in the reference.
Conflict of Interest. The author declares that there is no conflict of interests.

REFERENCES

[1] Singh, G., & Solanki, A. (2016). An algorithm to transform natural language into sql queries for relational databases. Selforganizology, 3(3), 100-116. Sripad, Joshi, and Laxmaiah E. n.d. 2013. Survey of Natural Language Interface to Databases.
[2] Kim, H., So, B. H., Han, W. S., & Lee, H. (2020). Natural language to SQL: Where are we today? *Proceedings of the VLDB Endowment*, *13*(10), 1737-1750.
[3] Vig, Jesse, and Kalai Ramea. "Comparison of transfer-learning approaches for response selection in multi-turn conversations." Workshop on DSTC7. 2019.
[4] Yu, Tao, et al. "Syntaxsqlnet: Syntax tree networks for the complex and cross-domain text-to-SQL task." arXiv preprint arXiv:1810.05237 (2018).
[5] Sun, Zeyu, et al. "A grammar-based structural CNN decoder for code generation." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. 2019.
[6] Finegan-Dollak, Catherine, et al. "Improving text-to-SQL evaluation methodology." arXiv preprint arXiv:1806.09029 (2018).
[7] Yu, Tao, 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).
[8] Hwang, Wonseok, et al. "A comprehensive exploration on wikisql with table-aware word contextualization." arXiv preprint arXiv:1902.01069 (2019).
[9] Lin, Kevin, et al. "Grammar-based neural text-to-SQL generation."arXiv preprint arXiv:1905.13326 (2019).

[10] Maas, Andrew, et al. “Learning word vectors for sentiment analysis.” Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies. 2011.
[11] Xu, Xiaojun, Chang Liu, and Dawn Song. “Sqlnet: Generating structured queries from natural language without reinforcement learning.” arXiv preprint arXiv:1711.04436 (2017).
[12] Gardner, Matt, et al. “Allennlp: A deep semantic natural language processing platform.” arXiv preprint arXiv:1803.07640 (2018).
[13] Affolter, Katrin, Kurt Stockinger, and Abraham Bernstein. “A Comparative Survey of Recent Natural Language Interfaces for Databases.” The VLDB Journal 28.5 (2019): 793–819. Crossref. Web.
[14] Sujatha, B., & Raju, S. V. (2016). Natural Language Query Processing for Relational Database using EFFCN Algorithm. International Journal of Computer Sciences and Engineering, 4, 49-53.
[15] Sukthankar, N., Maharnawar, S., Deshmukh, P., Haribhakta, Y., & Kamble, V. (2017). nQuery-A Natural Language Statement to SQL Query Generator. In Proceedings of ACL 2017, Student Research Workshop (pp. 17-23).
[16] Stefan W., Ellen R., Gabriele S., (1996). Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing, Springer.
[17] T. Ono, H. Hishigaki, A. Tanigami, T. Takagi, (2001), Automated extraction of information on proteinprotein interactions from the biological literature, Bioinformatics. doi:10.1093/bioinformatics/17.2.155.
[18] Warren, D. H., & Pereira, F. C. (1982). An efficient easily adaptable system for interpreting natural language queries. Computational Linguistics, 8(3-4), 110-122.
[19] Woods, William A, Ronald M Kaplan, and Bonnie Nash-Webber. (1972) The lunar sciences natural language information system. Bolt, Beranek and Newman, Incorporated.
[20] Xu, X., Liu, C., & Song, D. (2017). Sqlnet: Generating structured queries from natural language without reinforcement learning. arXiv preprint arXiv:1711.04436.
[21] Yossi Shani, Tal Cohen, and Yossi Vainshtein. (2016) "Natural Language Interface for Databases." KUERI.ME. 2016. http://kueri.me/.
[22] Yaghmazadeh, N., Wang, Y., Dillig, I., & Dillig, T. (2017). Sqlizer: Query synthesis from natural language. Proceedings of the ACM on Programming Languages, 1(OOPSLA), 63.
[23] Zhong, V., Xiong, C., & Socher, R. (2017). Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. arXiv preprint arXiv:1709.00103.
[24] Lin, K., Bogin, B., Neumann, M., Berant, J., & Gardner, M. (2019). Grammar-based neural text-to-sql generation. *arXiv preprint arXiv:1905.13326*.
[25] Zhang, Rui, et al. “Editing-Based SQL Query Generation for Cross-Domain Context-Dependent Questions.” arXiv preprint arXiv:1909.00786 (2019).
[26] Wang, Bailin, et al. “Rat-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers.” arXiv preprint arXiv:1911.04942 (2019).

[27] Dar, Hafsa Shareef, et al. "Frameworks for Querying Databases Using Natural Language: A Literature Review." arXiv preprint arXiv:1909.01822 (2019).
[28] Popescu, A. M., Etzioni, O., & Kautz, H. (2003, January). Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces* (pp. 149-157)
[29] Uma, M., Sneha, V., Sneha, G., Bhuvana, J., & Bharathi, B. (2019, February). Formation of SQL from natural language query using NLP. In *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)* (pp. 1-5). IEEE
[30] Sukthankar, N., Maharnawar, S., Deshmukh, P., Haribhakta, Y., & Kamble, V. (2017, July). nQuery-A natural language statement to SQL query generator. In *Proceedings of ACL 2017, Student Research Workshop* (pp. 17-23)
[31] Montgomery, C. A. (1972, August). Is natural language an unnatural query language? In *Proceedings of the ACM annual conference-Volume 2* (pp. 1075-1078)
[32] Iqbal, R., Murad, M. A. A., Selamat, M. H., & Azman, A. (2012, March). Negation query handling engine for natural language interfaces to ontologies. In *2012 International Conference on Information Retrieval & Knowledge Management* (pp. 249-253). IEEE.
[33] Mukherjee, P., Chattopadhyay, A., Chakraborty, B., & Nandi, D. (2021). Natural language query handling using extended knowledge provider system. *International Journal of Knowledge-based and Intelligent Engineering Systems*, *25*(1), 1-19
[34] Huang, P. S., Wang, C., Singh, R., Yih, W. T., & He, X. (2018). Natural language to structured query generation via meta-learning. *arXiv preprint arXiv:1803.02400*
[35] Small, D. W., & Weldon, L. J. (1983). An experimental comparison of natural and structured query languages. *Human Factors*, *25*(3), 253-263
[36] Koutrika, G., Simitsis, A., & Ioannidis, Y. E. (2010, March). Explaining structured queries in natural language. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)* (pp. 333-344). IEEE
[37] Gur, I., Yavuz, S., Su, Y., & Yan, X. (2018, July). Dialsql: Dialogue based structured query generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1339-1349)
[38] Kaplan, S. J. (1984). Designing a portable natural language database query system. *ACM Transactions on Database Systems (TODS)*, *9*(1), 1-19
[39] Yaghmazadeh, N., Wang, Y., Dillig, I., & Dillig, T. (2017). SQLizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, *1*(OOPSLA), 1-26
[40] Yaghmazadeh, N., Wang, Y., Dillig, I., & Dillig, T. (2017). SQLizer: query synthesis from natural language. *Proceedings of the ACM on Programming Languages*, *1*(OOPSLA), 1-26
[41] Androutsopoulos, I., Ritchie, G. D., & Thanisch, P. (1995). Natural language interfaces to databases–an introduction. *Natural language engineering*, *1*(1), 29-81
[42] Kate, A., Kamble, S., Bodkhe, A., & Joshi, M. (2018, March). Conversion of natural language query to SQL query. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)* (pp. 488-491). IEEE.

[43] Song, Y., Wong, R. C. W., Zhao, X., & Jiang, D. (2022). Speech-to-SQL: Towards Speech-driven SQL Query Generation from Natural Language Question. *arXiv preprint arXiv:2201.01209.*
[44] Sujatha, B., & Raju, S. V. (2014). A Flexible and Efficient Natural Language Query interface to databases. *International Journal of Computer Science and Information Technologies*, *5*(5), 6464-6467.
[45] Dekleva, S. M. (1994). Is natural language querying practical? *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, *25*(2), 24-36.
[46] Narechania, A., Fourney, A., Lee, B., & Ramos, G. (2021, April). DIY: Assessing the correctness of natural language to sql systems. In *26th International Conference on Intelligent User Interfaces* (pp. 597-607).
[47] Blanning, R. W. (1986). A System for natural language communication between a decision model and its users. *IFAC Proceedings Volumes*, *19*(17), 77-85.
[48] Amble, T. (2000, April). BusTUC-a natural language bus route oracle. In *Sixth Applied Natural Language Processing Conference* (pp. 1-6).
[49] Narechania, A., Fourney, A., Lee, B., & Ramos, G. (2021, April). DIY: Assessing the correctness of natural language to sql systems. In *26th International Conference on Intelligent User Interfaces* (pp. 597-607).
[50] Zhang, X., Cheng, G., & Qu, Y. (2007). Ontology summarization based on rdf sentence graph. WWW '07.