Structuring Natural Language to Query Language: A Review

B. Nethravathi
Department of Information Science and Engineering
JSS Academy of Technical Education, Bangalore
Karnataka, India
nethravathi.sai@gmail.com

G. Amitha
Department of Information Science and Engineering
JSS Academy of Technical Education, Bangalore
Karnataka, India
amithagk98@gmail.com

Anusha Saruka
Department of Information Science and Engineering
JSS Academy of Technical Education, Bangalore
Karnataka, India
anushasaruka21@gmail.com

T. P. Bharath
Department of Information Science and Engineering
JSS Academy of Technical Education, Bangalore
Karnataka, India
bharathtp.1js16is018@gmail.com

Setu Suyagya
Department of Information Science and Engineering
JSS Academy of Technical Education, Bangalore
Karnataka, India
setu.suyagya@gmail.com

***Abstract*-SQL (Structured Query Language) is a structured language for specialized purposes used to communicate with the data stored in a database management system. It uses dynamic and sophisticated query commands for processing and controlling data in a database, which can become an obstacle for users with no previous experience. In order to address this constraint, we have analyzed the existing models in Natural Language Processing, which convert a native-language query into an SQL query. Thus, any novice user can use the SQL program and eliminate the need to generate any complex queries. This work is a detailed survey of the existing literature.**

Keywords-structured query language; natural language processing; query

I. INTRODUCTION

Structured data are useless without the use of a data management system. Such a system makes the information organized making easier the communication between it and its users. In order to communicate effectively, one must have complete knowledge of database languages such as the Structured Query Language (SQL), converting natural language queries in machine-translated queries. Although SQL was developed primarily for commercial users, even the most capable ones find difficulties in creating queries, due to the large amount of data required to fully define a modern database. A common solution to overcome these barriers is the addition of natural language databases. This enables novice users to submit their queries in any language to access the required information.

Corresponding author: B. Nethravathi

II. LITERATURE SURVEY

Natural language in databases allows users to have easy approach to access the data, without the knowledge of a query language. The system in [1] is an idea driven program consisting of local language queries of natural language. A two-part approach has been proposed: the Natural Language Query (NLQ) is translated into the Object Query Language (OQL) first and then the query is converted to SQL. The two-step approach will allow the users to physically extract data from the semantics of the question, which gives physical independence. In addition, ontologies that provide rich semantics, are missing from the related schema. The system achieves 100% accuracy on the MAS and GEO function, and 99% accuracy on the FIN load factor used for complex ontology. In addition, ATHENA gains 87.2%, 88.3%, and 88.9% recall in the GEO, MAS, and FIN activities. Natural Language Interface to DataBase (NLIDB) has problems regarding language (language similarity) and domain behavior. The actual problem with NLIDBs is sequential translation. Stack Exchange Data Explorer is a website that can be used to train the natural methods of internal information. It is reported [2] that it promotes basic results in small hand-held manuals. Although the SQL output can occasionally be ineffective and is hardly a desirable SQL query, the results are encouraging.

The main aim of data centers is allowing business analysts to make superior accord. Technology has grown and repositories have proved to be increasingly useful. Because of this, more data are being incremented to datasets and their schemas become more compound. These systems tend to make inefficient match for incompetent users who need answers to unexpected crude questions. An important solution is to use a graph pattern similar to an algorithm which uses a metadata database model. Actions with original data show that SODA [3] generates queries with high preciseness, and makes it way easier to use devices that store abundant data. The main aim of SODA is to generate SQL statements built on simple keywords using the algorithm's graphical pattern. The experiments show that queries that are performed with great accuracy are reminiscent of handwritten queries. The key issue of SODA is that it can confuse the definition of words when it comes to the association and inheritance relations between adjacent tables. In addition, SODA allows reducing the noises' power in the data or schema and the quality of data issues by upgrading the appropriate graph or by broadening the graph algorithm. BANKS [4] is a program that provides search based on keywords in databases. BOARDS enables novice users to get access to data or extract information from a database without any SQL or programming knowledge. The BANKS models depend on graphs and are connected by links inserted by an external button. Questions are used as focus trees to connect keywords in the generated SQL query. Combining browsing and keyword identification for related information, BANKS allows inexperienced users to query and browse a database easily. BOARDS dramatically reduces the effort involved in publishing web-related information and making it searchable.

The approach in [5] promises accuracy, completeness and preciseness of a NLI. Their model could be improvised to analyze a broad spectrum of questions in analytical and theoretical sense. The highlight of this approach is the use of the max-flow algorithm which collects solutions and also disregards those solutions that disobey semantic constraints. The proposed technology in [6] addresses challenges like the lack of good visual interactive tools and the fact that NLIDBs have still not made sufficient progress into practical and commercial products. This model has a considerable leverage on DNN models as the core of its NL interface system. SQLNet [7] generates SQL queries from natural language with no reinforcement learning. It tends to cater challenges like issues of sequence-to-sequence models with respect to different queries with similar outputs. It is mainly designed to be universal, in such a way that each and every SQL query can be generated. The proposed system is known to have employed a sequence-to-set model in order to generate SQL queries when the order is insignificant. This model has shed light on novel solutions to problems on structural generation. The proposed system in [8] mainly aimed at solving the problem of insufficient schema or query knowledge of users by slimming down the number of all possible cogent predicates for a particular question in hand. A semantic parser that could scale to freebase rather depending on annotated logical forms was trained. This work interacted successfully with different streams of work, one of which involved learning models of semantics and another which involved connecting natural language and open domain databases. The proposed system for "Conversion of Natural Language Query to SQL Query" allows a novice user to use text or speech to retrieve data. Also, provides an additional feature where the user can insert, update or delete data from the database. The text or speech is processed in succession. This system uses four main steps to convert a natural language query to an accurate SQL query (Tokenization, Lexical Analysis, Syntactic Analysis, and Semantic Analysis) [4]. The system in [9] enables the novice user to input queries in Hindi. This system uses analyzers such as word group analyzer morphological analyzer, which excerpts the keyword out of the query. It then uses a pattern matching formula to categorize the keyword. A domain-specific dictionary is used to retrieve the semantics. To each of these keywords, SQL grammar is mapped and a proper syntactic query is generated. The proposed technique in [10] uses simple natural language (English) that has basic POS such as verbs, nouns, adjectives, and adverbs, which are translated into an SQL query. This is followed by a preprocessing step which involves processes such as tokenization, tagging POS, and stemming. Then, it is processed with semantic disambiguation by categorizing the words based on their domain and replacing them with corresponding SQL keywords. Then these keywords are clubbed into proper syntax and a query is generated.

The proposed system in [11] mainly focuses on eliminating the difficulties of matching Natural Language to SQL queries. The main adversities faced in this conversion are join path inference and keyword mapping. Keyword word mapping involves retrieving candidate mappings, scoring, and pruning, ranking configurations. Join path inference mainly deals with joins between the schema: Since the users won't be aware of the internal structure of the schema, they won't know their connectivity. So, join path inference involves generating join paths, scoring join paths, and self-joins. Authors in [12] propose a novel, fully automated method for producing SQL queries from the input given in natural language. Their method first creates a query sketch or partial query of the input using standard NLP procedures. This sketch is then passed to a type-directed program which generates a complete SQL query. To further improve accuracy, this method performs fault localization when there is an error and database repair knacks to improve the original sketch. These steps are repeated in a confidence driven refinement loop and the top results are presented to the user. Authors in [13] aimed to reconnect the void between natural language and query language with nominal domain knowledge. Neural networks were used to encode and decode the input parameters and their machine representations. This method is described in two models: Sequence-to-Sequence and Sequence-to-Tree. The method has two steps, an encoder to convert the natural language input to vector representation and to form a proper SQL query using a decoder from this vector. It also includes an attention layer which enables the model to learn soft alignments between inputs and outputs and present a way to handle rare mentions of entities and numbers. This approach demonstrates the same or better performance with similar models though it doesn't use representation specific features. Authors in [14] proposed an improved semantic parser named SCISSOR that generates a sophisticated meaning representation language from a natural language sentence. Their approach uses two target meaning representation languages (MRLs). They have created a semantic parsing framework that constructs a Semantically

Augmented Parse Tree (SAPT) and uses a recursive procedure to construct Meaning Representations (MRs) for each node. The experimental results demonstrate that this approach produces more accurate MRs than several previous methods. A detailed representation of the reviewed existing work is depicted in Table I.

TABLE I. DETAILED ANALYSIS

Ref.	Defined problems	Conclusion and future work
[1]	Understanding the semantics of the query is a challenging task in the development of NLIDBs.	It is an improvement of the previous attempts at the utilization of database schemas as the skeleton of the system. It provides physical independence, by separating logical and physical schemas. Future work: Extend ATHENA to include natural language dialogue capabilities and to handle self-join queries through nesting
[2]	Linguistic uncertainties, domain inflexibility, non availability of model benchmarks in data driven methods which lead to additional complexity.	This design is the first large corpus for data driven neural natural language interfaces to databases and has succeeded in training an E2E NLIDB using sequence to sequence model. A future scope is the implementation of a stacked decoder to generate a query sketch.
[3]	High complexity of data-warehouses which clashes with non-tech users.	Uses relationships of inheritance to eliminate the ambiguity in the meaning of words. Future work: Debate the effects of the usage of DBPedia for pairing keyword queries against numerous synonyms found by the model.
[4]	A substantial increase in the number of online database users. Failure of previous attempts at introducing simple query languages.	Successful implementation using servlets, JavaDataBase Connectivity to an existing universal database of IBM. With proper parameter settings, the system in most cases outputs the more initiative solutions before the less initiative ones.
[5]	Need of more NLIs, low accuracy of previous NLIs, and people unwilling to switch over to NLIs from previous interfaces	This is supposedly the first formal fulfilment of soundness and completeness of a natural language interface. Their method could be extended to include large classes of questions.
[6]	Lack of good visual interactive tools for databases, non availability of commercial and practical NLIDBs.	This approach makes an adequate use of DNN models as the core of its natural language interface system. Future scope: To enable users to incrementally generate queries in a chatbot-like interface, where the system can question the user for clarifications and use prediction tasks.
[7]	Issues of sequence-to-sequence model regarding different queries with identical output. Previous methods of reinforcement learning only bring about a small increment in performance.	Employs a SEQ2SET model to form SQL queries with no order, and a column attention mechanism, which can further improve the model's execution and efficiency. This approach shines new light on innovative solutions to structural generation problems with no order.
[8]	Reducing the large number of logical implications for a given natural language input.	This model intersects with two other techniques: The first includes attaining knowledge about models of semantics influenced by denotations or interactions with the external factors and the second involves connecting NLs and open-domain DBs.
[9]	Strict syntax and complexity of SQL queries. Inability of novice users to use SQL efficiently.	This model enables users to use English language to retrieve data from a database, thus eliminates any requirement to have knowledge about any complex query language.
[10]	Addition of natural language interfaces to databases has become promising. Lack of NLIs in Indian languages.	Successfully generates SQL queries from Hindi sentences. It makes use of a morphological analyzer and a word group analyzer to obtain and mark keywords. This model uses domain-oriented dictionaries to determine the keyword type. In the future, this approach can be improved to a multilingual system.
[11]	Aims to make the conversion from natural language to query language automatic and to put more emphasis on requirement specification.	Aims to increase the performance of query generation from natural language queries by preventing ambiguities in meaning of words. The future scope of this approach is to extract OO data from software specification.
[12]	The semantic difference between an NL query and relevant data is an overbearing problem in the creation of NLIs to databases.	The performance of existing NLIs is improved by this method using information in the SQL query log of a data structure. Future scope: To research the influence of additional data in the SQL query log such as user sessions
[13]	Lack of knowledge about forming correct queries. Importance of familiarity with the database schema in order to retrieve data.	The parse-synthesize-repair loop approach can be instantiated with multiple different settings to generate programs from an NL.
[14]	High dependency on advanced vocabulary, custom made templates, and language specific features which are either representation or domain-specific	Outperforms older approaches despite not having domain or representation specific features. This model can also be applied to similar structured prediction problems such as constituency parsing.
[15]	Shallow analysis in previously used semantic methods.	By developing a state-of-the-art statistical parsing model containing semantic data, it is able to have syntactic and semantic clues to develop a robust interpretation that supports the creation of complete formal meaning representations.

III. METHODOLOGY

The solution to the major challenge of removing the middle layer hurdles discussed above is NLIDBs. There are several inter-related domains to serve this purpose, some of which are [15]:

- Natural Language Processing
- Database implementation style
- Compiler design

There are mainly two known options for converting natural language to SQL [15]:

- Semantic Parsing: The given input is translated to a standard not generalized format which depends on several factors such as the quality of lexicons, realm representation-specific features, etc.
- Neural Networks: Instead of using hard coded rules, this method relies on self-improvising and cognitive techniques to drive the task of conversion.

The next important step is obtaining data from DBMS. Two paths can be considered [16]:

- Keyword-based methods: Although comprehensibility is an advantage, processing large sentences can be difficult.
- Structured query method: This method is expressive and powerful but complex to use.

It is challenging to meet the requirements because converting the natural language input into a query of schema structure is a complex task. One proposed solution is NaLIR (Natural language Interface working with RDBMS) [15]. Another perspective to cater to the challenge of NL input conversion is with respect to a graph database system [16]. This perspective most importantly fairs well to carry out computations when there are inter-related data. Unlike a conventional RDBMS, this system is entirely structured on inter-relationships between data by treating them not as a schema structure but as data, just like other values. There are four main steps involved in the conversion of natural language input to SQL queries [18-25].

A. Tokenization

Tokenization is the process of demarcating and classifying all possible sections of a string of input characters. The tokens that result from this process are then passed on to some other form of processing. This step can be considered as a sub-task of the parsing input. Tokenization is followed by Parsing. From this point, the data that are already interpreted may be loaded into respective data structures for further interpretation, compiling or general use.

B. Parsing

The challenges that need to considered while parsing are [17]:

- Dependences of some intermediate representations on the database language.
- Domain-dependent syntactic parsing.
- No/lack of any improvement in the waiting-time translating already processed input.

There are two types of syntactic rules to be followed to parse an NL query in the proposed system [17], where the NL query is first syntactically parsed and then evaluated against DBMS using machine learning. These two types of rules are:

- Rules that link only non-terminal symbols (non-leaf nodes).
- Rules that link only terminal symbols (leaf nodes).

Natural Language Query Definitions [17], is a system proposed to represent all classes of logically interpreted natural language queries, as a solution to minimize the waiting-time for translating the already translated input.

C. Lexical Analysis

The next step in the process of conversion of a NL query is lexical analysis. This step involves analyzing a stream of characters into a lexicon token sequence which is further fed to the parser. This process can be seen as a rough equivalent of splitting text written in a natural language into a sequence consisting of words and punctuation symbols. Although tokenization can be handled by the parser, we bother with tokenization to make the parser simpler and ensure that it is decoupled from the character encoding which is used for the source code.

D. Syntactic Analysis

The main task involved in this step is analyzing the syntactic structure of the program and its components and to check for errors. This step ensures that the dictionary of table names, keywords, and attributes is maintained. Every tokenized word is mapped with the respective attributes in the maintained dictionary. It is passed onto the next step for further analysis.

E. Semantic Analysis

Semantic analysis essentially ensures that the statements and declarations of a program are correct in the semantic sense. This makes sure that their meaning is consistent with the control structures and data types. This step involves:

- Label checking: Ensures that the label references in a program exist.
- Type checking: Ensures that the data types are used in consistence with their definition.
- Flow control checking: Ensures that the control structures are used in their proper fashion.

Semantic analysis is a collection of procedures, not a separate module in the compiler, which is called by the parser at appropriate times as per grammar requirement.

IV. CONCLUSION

Communication with databases using the standard natural language is still problematic. The planned future system will be useful for many people, but the main aim is to address it directly to T&P managers who are comfortable with the simple English language. The planned program aims to eliminate the need for any knowledge about database languages such as SQL. In the system, natural speech will be treated as input, making it more desirable. Also, the proposed system provides an easy-to-use graphical user interface, which will facilitate access.

REFERENCES

[1] D. Saha, A. Floratou, K. Sankaranarayanan, U. F. Minhas, A. R. Mittal, and F. Özcan, "ATHENA: an ontology-driven system for natural language querying over relational data stores," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 1209–1220, Aug. 2016, https://doi.org/10.14778/2994509.2994536.

[2] F. Brad, R. C. A. Iacob, I. A. Hosu, and T. Rebedea, "Dataset for a Neural Natural Language Interface for Databases (NNLIDB)," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Taipei, Taiwan, Nov. 2017, pp. 906–914, Accessed: Nov. 28, 2020. [Online]. Available: https://www.aclweb.org/anthology/I17-1091.

[3] L. Blunschi, C. Jossen, D. Kossman, M. Mori, and K. Stockinger, "SODA: Generating SQL for Business Users," *arXiv:1207.0134 [cs]*, Jun. 2012, Accessed: Nov. 28, 2020. [Online]. Available: http://arxiv.org/abs/1207.0134.

[4] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword searching and browsing in databases using BANKS," in *Proceedings 18th International Conference on Data Engineering*, San Jose, CA, USA, Feb. 2002, pp. 431–440, https://doi.org/10.1109/ICDE.2002.994756.

[5] A.-M. Popescu, O. Etzioni, and H. Kautz, "Towards a theory of natural language interfaces to databases," in *Proceedings of the 8th international conference on Intelligent user interfaces*, New York, NY, USA, Jan. 2003, pp. 149–157, https://doi.org/10.1145/604045.604070.

[6] F. Basik *et al.*, "DBPal: A Learned NL-Interface for Databases," in *Proceedings of the 2018 International Conference on Management of Data*, New York, NY, USA, May 2018, pp. 1765–1768, https://doi.org/10.1145/3183713.3193562.

[7] X. Xu, C. Liu, and D. Song, "SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning," *arXiv:1711.04436 [cs]*, Nov. 2017, Accessed: Nov. 28, 2020. [Online]. Available: http://arxiv.org/abs/1711.04436.

[8] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic Parsing on Freebase from Question-Answer Pairs," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA, Oct. 2013, pp. 1533–1544.

[9] A. Kate, S. Kamble, A. Bodkhe, and M. Joshi, "Conversion of Natural Language Query to SQL Query," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India, Mar. 2018, pp. 488–491, https://doi.org/10.1109/ICECA.2018.8474639.

[10] R. Kumar and M. Dua, "Translating controlled natural language query into SQL query using pattern matching technique," in *International Conference for Convergence for Technology-2014*, Pune, India, Apr. 2014, pp. 1–5, https://doi.org/10.1109/I2CT.2014.7092161.

[11] A. Iftikhar, E. Iftikhar, and M. K. Mehmood, "Domain specific query generation from natural language text," in *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*, Dublin, Ireland, Aug. 2016, pp. 502–506, https://doi.org/10.1109/INTECH.2016.7845105.

[12] C. Baik, H. V. Jagadish, and Y. Li, "Bridging the Semantic Gap with SQL Query Logs in Natural Language Interfaces to Databases," *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 374–385, Apr. 2019, https://doi.org/10.1109/ICDE.2019.00041.

[13] N. Yaghmazadeh, Y. Wang, I. Dillig, and T. Dillig, "SQLizer: query synthesis from natural language," *Proceedings of the ACM on Programming Languages*, vol. 1, no. OOPSLA, p. 63:1–63:26, Oct. 2017, https://doi.org/10.1145/3133887.

[14] L. Dong and M. Lapata, "Language to Logical Form with Neural Attention," *arXiv:1601.01280 [cs]*, Jun. 2016, Accessed: Nov. 28, 2020. [Online]. Available: http://arxiv.org/abs/1601.01280.

[15] R. Ge and R. Mooney, "A Statistical Semantic Parser that Integrates Syntax and Semantics," in *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, Ann Arbor, Michigan, Jun. 2005, pp. 9–16, Accessed: Nov. 28, 2020. [Online]. Available: https://www.aclweb.org/anthology/W05-0602.

[16] D. Shah and D. Vanusha, "Optimizing Natural Language Interface for Relational Database," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 4, pp. 131–135, Apr. 2019.

[17] C. Sun, "A Natural Language Interface for Querying Graph Databases," M.S. thesis, Massachusetts Institute of Technology, 2018.

[18] H. Bais, M. Machkour, and L. Koutti, "A Model of a Generic Natural Language Interface for Querying Database," *International Journal of Intelligent Systems and Applications*, vol. 8, no. 2, pp. 35–44, Feb. 2016, https://doi.org/10.5815/ijisa.2016.02.05.

[19] J.-H. Tao, J. Huang, Y. Li, Z. Lian, and M.-Y. Niu, "Semi-supervised Ladder Networks for Speech Emotion Recognition," *International Journal of Automation and Computing*, vol. 16, no. 4, pp. 437–448, Aug. 2019, https://doi.org/10.1007/s11633-019-1175-x.

[20] B.-T. Zhang, X.-P. Wang, Y. Shen, and T. Lei, "Dual-modal Physiological Feature Fusion-based Sleep Recognition Using CFS and RF Algorithm," *International Journal of Automation and Computing*, vol. 16, no. 3, pp. 286–296, Jun. 2019, https://doi.org/10.1007/s11633-019-1171-1.

[21] N. R. Nayak, P. K. Dash, and R. Bisoi, "A Hybrid Time Frequency Response and Fuzzy Decision Tree for Non-stationary Signal Analysis and Pattern Recognition," *International Journal of Automation and Computing*, vol. 16, no. 3, pp. 398–412, Jun. 2019, https://doi.org/10.1007/s11633-018-1113-3.

[22] H. Sasaki, S. Yamamoto, A. Agchbayar, and N. Nkhbayasgalan, "Extracting Problem Linkages to Improve Knowledge Exchange between Science and Technology Domains using an Attention-based Language Model," *Engineering, Technology & Applied Science Research*, vol. 10, no. 4, pp. 5903–5913, Aug. 2020, https://doi.org/10.48084/etasr.3598.

[23] S. Khalid and S. Wu, "Supporting Scholarly Search by Query Expansion and Citation Analysis," *Engineering, Technology & Applied Science Research*, vol. 10, no. 4, pp. 6102–6108, Aug. 2020, https://doi.org/10.48084/etasr.3655.

[24] M. Alsuwaiket, A. H. Blasi, and K. Altarawneh, "Refining Student Marks based on Enrolled Modules' Assessment Methods using Data Mining Techniques," *Engineering, Technology & Applied Science Research*, vol. 10, no. 1, pp. 5205–5210, Feb. 2020, https://doi.org/10.48084/etasr.3284.

[25] S. R. Basha and J. K. Rani, "A Comparative Approach of Dimensionality Reduction Techniques in Text Classification," *Engineering, Technology & Applied Science Research*, vol. 9, no. 6, pp. 4974–4979, Dec. 2019, https://doi.org/10.48084/etasr.3146.