The history and recent advances of Natural Language Interfaces for Databases Querying

Khadija Majhadi[1*], and *Mustapha* Machkour[1]

[1]Team of Engineering of Information Systems, Faculty of Sciences Agadir, Morocco

Abstract. Databases have been always the most important topic in the study of information systems, and an indispensable tool in all information management systems. However, the extraction of information stored in these databases is generally carried out using queries expressed in a computer language, such as SQL (Structured Query Language). This generally has the effect of limiting the number of potential users, in particular non-expert database users who must know the database structure to write such requests. One solution to this problem is to use Natural Language Interface (NLI), to communicate with the database, which is the easiest way to get information. So, the appearance of Natural Language Interfaces for Databases (NLIDB) is becoming a real need and an ambitious goal to translate the user's query given in Natural Language (NL) into the corresponding one in Database Query Language (DBQL). This article provides an overview of the state of the art of Natural Language Interfaces as well as their architecture. Also, it summarizes the main recent advances on the task of Natural Language Interfaces for databases.

1 Introduction

For several years, databases have become the preferred medium for data storage in all information management systems. It has been an active research topic for a long time, especially since there is always a need for database users to automatize the process of accessing data on the database. The extraction of this data requires prior knowledge of a language called Database Query Language (DBQL), such as SQL (Structured Query Language). However, this is an honest limitation for non-expert users who do not have the technical skills to write such queries. Several solutions have been proposed to face this problem [1]. One of these solutions is the use of Natural Language to interact directly with a database.

Natural Language Processing (NLP) is one of the most challenging areas in Artificial Intelligence (AI) research located at the intersection of Computer Science, AI, and Linguistics [2]. NLP is used in Human-Computer Interaction for information retrieval, machine translation, and linguistic analysis. Natural Language Interface to Database (NLIDB) is one of the traditional applications of the NLP domain. It is a powerful example of Question answering systems (QAS) for querying structured and unstructured databases. NLIDB has been an interactive area of research that aims to provide accurate answers to user questions expressed in Natural Language and generalize access to databases for different types of users regardless of their technical skills. It is one of the fundamental subjects of artificial intelligence and databases [3].

The main objective of this paper is to present the history of the NLIDB systems and provide an overview of recent Natural Language Interfaces for databases by highlighting the architectures they used before concluding and giving some perspectives on our future work.

2 Natural Language Interfaces to Databases (NLIDB)

2.1 Definition

Natural Language Interface has been an interesting area of research. NLIDB is an intelligent and flexible system that has already appeared in the late 1960s and early 1970s. It can translate a request in Natural Language into a request in the database query language. So, users can interact with the database more conveniently and flexibly.

There have been different approaches and architectures in the field of NLIDBs: [4]

a. Pattern matching systems: Pattern Matching systems appeared in the late 60s and early 70s. It uses the FLIP (Formal List Processor) language based on the structure of the LISP (List Processing) language. It creates a query in NLIDB from an entry corresponding to a predefined model using a set of rules. Many NLIDB systems use Pattern matching systems because of its simplicity to map user inputs expressed in natural language (NL) to a query in database query language: the NL query is processed by first associating it to a model. Then it is transformed into a logical form according to the model to which it belongs. Finally, the system formulates the query in NLIDB. Moreover, these

* Corresponding author: khadija.majhadi@gmail.com

systems often manage to find reasonable answers to user requests. But, they are limited just to a specific database. One of the best systems based on this architecture is ELIZA.

b. Syntax-based systems: In syntax-based systems, the Natural Language query is parsed syntactically, and the resulting parsing tree is directly mapped to a Database Query Language expression. These systems use grammatical rules that describe the different possible syntactic structures of queries and a lexicon of words that may appear in the user's queries. The main advantage of this approach is that it provides detailed information about sentence structure, words, grammatical function, the relationships between these words, and how sentences can be grouped to form more complex sentences. The best known syntax-based NLIDB is LUNAR [5].

c. Semantic grammar systems: A semantic grammar system is similar to a syntax-based system. The query result is obtained by mapping the analysis tree to a database query [6]. The basic idea of such a system is to simplify the analysis tree as much as possible, by removing unnecessary nodes or combining some nodes. But, the main disadvantage is that the analysis tree in this system has specific structures and node labels, which makes it useless for other applications [7]. Many NLIDBs such as Planes, Ladder, and Rel are based on this architecture.

d. Intermediate Representation Languages: Intermediate representation systems were proposed because of the difficulties of translating user queries expressed in Natural Language (NLQ) directly into a database query language. The idea of this method is to first match the NLQ to an intermediate logical query expressed in a representation language. Then, it will be translated into a query in NLIDB. One known system that uses this architecture is PRECISE [8].

There are different ways of classifying NLIDB systems:

- Graphical NLIDBs: Presents users with an interface where they can make proposed selections for query formulation. NL-Menu system as an example [9].
- Textual NLIDBs: Allows users to express their queries in NL by directly writing it easily, LUNAR and PRECISE are systems that use Textual NLIDB.
- NLIDBs dependent on the database domain: it is necessary to know beforehand all the particularities of the field of application like LUNAR, ASK, and GILNDB systems.
- NLIDBs independent of the database domain: PRECISE is an example of this type of NLIDB system where knowledge of the field of application is not necessary.
- Classification of NLIDB by language: Most of the NLIDBs that exist answer English requests since this language is the main language of several countries. However, this does not prevent having other NLIDBs that allow access to the information stored in a database through the formulation of user queries in other languages: Arabic NLIDBs [10], Indien NLIDBs [11], French NLIDBs [12], Chinese NLIDBs [13], Bengali Language Query Processing System [14], and multilingual NLIDBs (the EDITE system which supports French, English, and Spanish languages).

2.2 Advantages and disadvantages of NLIDBs

The main advantages of NLIDBs are listed below:

- Absence of artificial language: NLIDB systems allow users to access the information stored in a database using queries written in Natural Language easily.
- Simple to use: The use of NLIDB is easier than the query languages of databases or forms [15].
- Better for some questions: some queries that involve negation or quantification can be easily expressed in Natural Language.
- Easy to use for multiple tables: queries that involve joins between tables have no difficulty for users, it is sufficient that the user types the query and the system displays the requested results.
- Tolerance for grammatical errors: most NLIDB systems support small grammatical errors. While the majority of computer systems require syntax rules.

Despite the development of many NLIDB systems, their use is not widespread due to a set of constraints, some of which are listed below:

- Lack of obvious language coverage: Some NLIDB systems cannot answer all questions expressed in Natural Language.
- Absence of explanations in case of failures: In the case of failures, some NLIDBs do not provide any explanation at all.
- Disappointment with user expectations: Individuals can be misled by the ability of an NLIDB system to process all their queries in Natural Language.
- Suitability of NLIDB for any user.

3 The history of NLIDB systems: literature survey

Over the past 50 years, many attempts have been made to create intelligent Natural Language Interfaces for querying databases. The first NLIDBs had appeared in the late sixties and early seventies and included two very well-known systems: BASEBALL (created to answer questions about baseball games that were played in the American League in that period) [16] and LUNAR (answers questions about rock samples brought back from the moon. It managed to answer 80% of the proposed queries without any errors). Both of these systems are NLIDBs dependent on the database domain and could not be easily reconfigured for use in other areas of the database [17]. At the end of the 1970s, several other NLIDB systems appeared: like PLANES (this system even managed to respond to incoherent or vague user requests) [18], LIFER/LADDER (it uses semantic grammar systems to analyze and then respond

to user requests) [19] and the RENDEZVOUS system [20] (developed in San José at the IBM laboratory, it helps users to ask or formulate their requests in case of ambiguity for analysis). In the mid-1980s, several NLIDBs emerged. CHAT-80 is one of the best-known systems for its effectiveness in this period. It uses Semantic grammar techniques to process user queries written in Natural Language. The major problem with this system is that it can only be used for a specific database domain [21]. This system has formed the basis of several other systems such as MASQUE, DIALOGIC which allow users to answer their requests very quickly. Research in this area of NLIDB continued in the 1990s. The majority of these contributions focused on querying Relational Databases using Natural Language (NL) instead of SQL. However, these systems are still designed for a specific field of application. Androutsopoulos et al have developed a system called MASQUE/SQL to answer any query written in English as a Natural Language that depends on commercial databases [22].

After 1990, interesting NLIDB approaches have been proposed, which have a major advantage regarding the operation of these NLIDB systems/interfaces independently of the database domain without any need for reconfiguration. PRECISE is a system developed at the University of Washington by Ana and al (2004) [23]. It targets relational databases and the language used to query the database is SQL. The PRECISE system is motivating because it combines linguistic and mathematical approaches to achieve complete independence of information, without any support or configuration. It is one of the first NLIDBs that used the analyzer as a plug-in, so it could be easily modified to employ the latest advantages in the field of analyzers. But, PRECISE suffers from the problem of managing nested structures. NALIX (Natural Language Interface for an XML Database) is a generic and interactive interface, developed at the University of Michigan by Yunyao Li et al. in 2006 [24]. The database used for this system is an XML database with 'Schema-Free XQuery' as the database query language. This language is primarily designed to retrieve information from XML databases and perform keyword searches. The main advantage of Schema-Free XQuery is that it finds automatically all given relationships to many key-words without mapping a query into the exact schema of the database. The process of translating the query into Natural Language is a three-step process: generation, validation, and translation of the analysis tree into an XQuery request. The NaLIR system (2014) is a generic Interactive Natural Language Interface for Querying Relational Databases. NaLIR can accept a logically complex English language sentence as query input to resolve ambiguous interpretations. There is also a system that is developed this year that allows user queries to be evaluated with high security: when ambiguities exist, the system generates multiple probable interpretations for the user. Next, so many systems were developed such as a system for querying the database using a Universal Natural Language Interface based on Machine Learning approach (2016) [25], An Arabic Natural Language Interface for Querying Relational Databases based on Natural Language Processing and Graph theory methods (2018) [26].

4 Architectural analysis of the most current NLIDB systems

The majority of the existing systems, as discussed in the previous section, are NLIDB systems dependent on the database domain and have developed for a specific use. A lot of efforts by researchers have been done in recent years to find a perfect solution to the problem of transforming queries expressed in Natural Language into Structured Query Language to get the predetermined information from the database.

With the rise of deep learning techniques, especially convolutional and recurrent neural networks (CNN and RNN), the most recent works used the encoder-decoder model, and semantic analysis approach. Current deep learning-based end-to-end systems use sequence-to-sequence architecture or a variation of it [27]. Seq2seq and Sketch-based are two major approaches to process the Text-To-SQL task. Seq2SQL encodes the text using an encoder/decoder (CNN or RNN) to get the semantic representation of the input text to decode it for generating the SQL statement [28]. These systems apply the necessary analysis to the input request without taking into consideration its structure neither the schema of the database and maintain deep learning techniques in its core elements. The next present some of the efficient models utilized in this field. Recently proposed architectures achieved more than 80% accuracy on the well-known Text-to-SQL benchmarks such as WikiSQL [29] and Spider [30].

PHOTON:
It takes the user's query and the database schema as input and apply deep learning techniques in its core elements. It enhanced the robustness of the task of SQL Query Formation for Database facings the non-translatable user input with 63% structure accuracy on the spider dataset. So, PHOTON attains a competing performance on the field of text to SQL benchmark [31]. But, the current PHOTON system is still a prototype, with very limited user interactions and functions.

PHOTON consists of (a) a powerful neural semantic parser: It uses a question encoder based on the BERT DB schema and a pointer-generating decoder with static SQL correctness check control. (b) A human-in-the-loop question corrector wish is a discriminative neural sequence editor which detects potential confusion span(s) in the input question and suggests possible corrections for the user to give rephrasing until a translatable input is given by the user or a maximum number of iterations are conducted (c) SQL query executor and (d) a natural language response generator. The robustness and effectiveness of the system are generalized by evaluating the performance of each module separately.

An information extraction approach text-to-SQL called **IE-SQL** [32] was proposed in this field. The unified BERT-based extraction model is operated to perceive all types of slot mentions presented in the input sentence. Then, a BERT-based linker maps the recognized columns to the table schema for incorporating executable SQL queries.

Another new neural network architecture based on the pre-trained BERT called **M-SQL** [33]. The extraction of column-based value is split into value extraction modules, and value column matching. M-SQL was evaluated on a more complicated TableQA dataset. M-SQL achieves state-of-the-art results on TableQA. M-SQL consists of three parts, encoder that used BERT-wwm-ext to better learn Chinese word vector representation, column representation, and several sub-models. The global M-SQL model consists of eight sub-models to predict the select and the where clause of the SQL statement. The overall architecture of the model is explained in the original paper. M-SQL improves the performance of single-table SQL generation. But, the problem of whether the query can be answered for a certain database has not been studied by the system.

DBPal [34] is also a new approach that improves the performance of existing deep learning models for natural language to SQL translation. It automatically generates synthetic training data to improve overall translation accuracy and increase robustness to linguistic variation. The system includes the training phase and runtime phase as shown in the following figure.

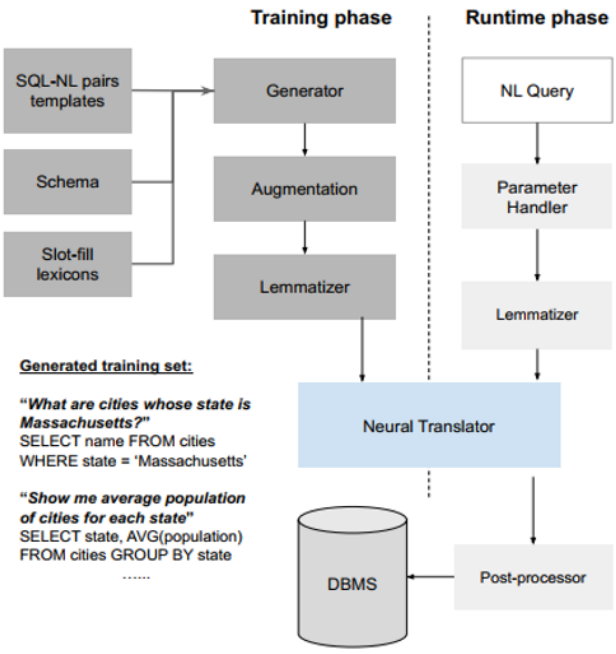

Fig. 1. DBPal system architecture.

RAT-SQL:

RAT-SQL [35] is a centralized framework wish Based on the relation-aware schema encoding and linking for Text-to-SQL parsers that use a self-attention mechanism to direct schema encoding, schema linking, and feature representation in a text-to-SQL encoder. The system can effectively encode more complex SQL queries that need joints and relationships within a non-ordered set of elements. This framework raises the exact match accuracy to only 57% to compare to PHOTON that has reached 63% on the same challenging Spider dataset. It represents first the database schema as a directed graph; its nodes are the columns and tables of the database schema. Then it applies relational self-attention to join global reasoning on schema entities and question words and then applies RAT-SQL to schema encoding and linking problems.

Bertrand-DR:

Bertrand-DR [36], a Discriminative-Reranker encoder-decoder based generative model, is a binary classifier that uses BERT to predict whether a given user's query is the right query for given utterance and schema information. First, the utterance and the query are encoded using BERT. It used SEP to separate utterance and the query, and Word-Piece to tokenize the speech. The tokens are combined to form the input token sequence. The token sequence is then encoded using BERT. The resulting encoding is passed through a linear classification layer and the model is trained using binary cross-entropy loss.

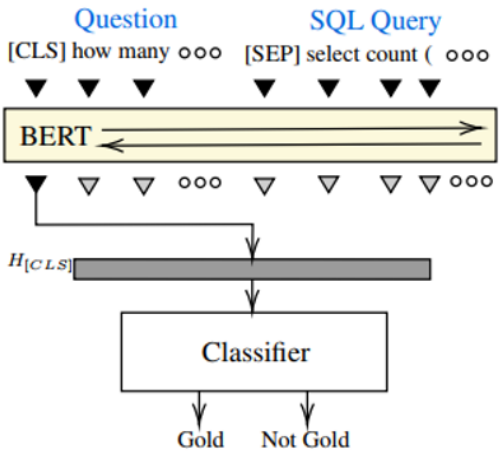

Fig. 2. Architecture overview of Bertrand-DR.

RYANSQL:

Recursively Yielding Annotation Network for SQL [37] is a neural network approach that Applies Sketch-based Slot Fillings for complex text-to-SQL in cross-domain databases. So considering the following sentence if we want to find out the names of STUDENTS who have either enrolled in 'SMI' or 'SMA', it can be done as:

Find the name of the student who has either enrolled in 'SMI' or 'SMA'

The SQL output of the previous nested query is:

Select NAME from STUDENTS where S_ID IN

(Select S_ID from STUDENTS_DEPARTMENT where D_ID IN

(SELECT D_ID from COURSE where D_NAME='SMA' or D_NAME='SMI'))

The nested SQL query is transformed into a set of non-nested SELECT statements. The system predicts and analyzes each SELECT statement separately. The input encoder consists of five layers. The input encoder; the process sketch-based slot-filling decoder is described in detail in the paper. The model could be improved by updating slot values based on other slots prediction results.

Another example of Natural Language Querying for Complex Nested SQL Queries is **ATHENA++** [38]. It combines linguistic patterns from NL queries with deep domain reasoning, using ontologies to capture the semantics of the domain schema on a new challenged Benchmark dataset called FIBEN. The overall architecture of the system contains Translation Index, Domain Ontology, Ontology to Database Mapping, and Query Translator.

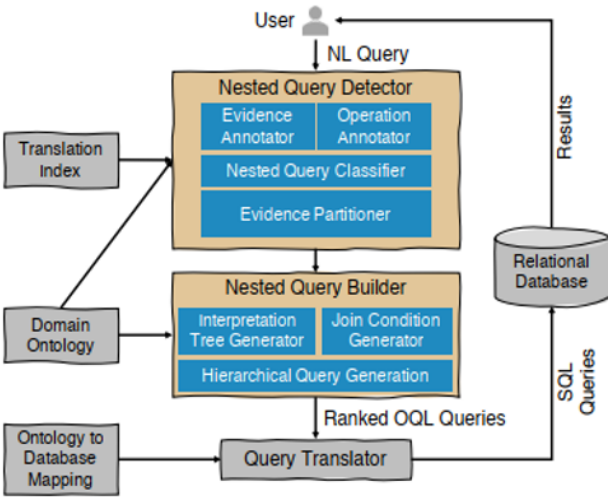

Fig. 3. The architecture of Athena++ system.

ValueNet:

ValueNet [39] is an end to end text-to-SQL system. The main idea of this approach is to use all the information on the base data as input for the neural network architecture. It's a new architecture sketch to extract values from a user question and predict the possible value candidates which are not specially mentioned in the question. Then a neural model is based on an encoder-decoder architecture used to synthesize the SQL query. The model is evaluated using the Execution accuracy on the Spider dataset (64%).

NLonSpark:

Natural Language on Spark [40] is implemented on top of Spark and Spark SQL which integrates relational functional programming API. It supports SQL queries and Hive Query Language. NLonSpark is based on NALIX system architecture. The Node Mapper communicates with an SQL concept abstraction layer that provides the functionality of an RDBMS. The SQL concept abstraction layer provides first the necessary schema information. Then, it provides indexing capabilities that Spark lacked, by implementing inverted indexes functionality on top of Apache Spark. The Spark-enabled Node Mapper achieved three operations: (1) Full-Text search (2) Columns summations: This functionality provides similarity amongst numerical columns and was implemented using the Dataframe API of Spark. (3) Column name similarity: This was provided by the SQL concept abstraction layer without calling Spark. The system could be improved to adapt NLonSpark to run on multi-application environments.

5 Evaluation

In this section, we provide some experimental results of recently developed systems highlighted in Sect. 4 based on the execution accuracy of the generated SQL queries on the development and the test, except the NLonSpark system evaluated in terms of scalability and impact of optimizations on YELP dataset. There are also different aspects on which a system can be evaluated (e.g., number of user interaction, ambiguity, efficiency, type of requests processed: simple or advanced queries, etc.), which we do not discuss in this paper. Each system category, based on its technical approach, has its strengths and weaknesses. But, since the current research included Deep Learning techniques in their translation process, the results are drastically improving. Table.1 shows the comparing results of the execution accuracy of each system.

Model	**Dataset**	**Accuracy (%)**	
		Dev	**Test**
RAT-SQL	Spider	62,7	57,2
Bertrand-DR	Spider	57,9	54,6
RYANSQL	Spider	66,6	58,2
M-SQL	TableQA	91,86	92,13
Photon	Spider	63,2	
TABERT	Spider	65,2	
ValueNet	Spider	64	
IE-SQL	WikiSQL	94,2	
Athena++	Spider	78,82	

Table 1. Execution accuracy of current NLIDB systems.

6 Conclusion

The objective of this survey is to present the state of the art of research with a long history of five decades that has been carried out in the field of Natural Language for Databases (NLIDB). NLIDB is a very active field in automatic language processing. Its purpose is to accept requests expressed in natural languages often used by non-technical users and to generate responses. It is a type of human-machine interface. Most of the tools developed are very efficient and have obtained encouraging results, but unfortunately, the majority of them have been developed so far for a specific use. These tools are developed only to be the interface of a database and are therefore exclusively compatible with it. Research in NLIDB is still in its infancy and needs to be continued. The use of NLIDB systems is not widespread and is not the optimal option for querying databases. This is mainly due to a large number of deficiencies in the NLIDB systems and the lack of a generic model that meets user expectations.

Our future work is to propose a model-based deep learning technique facing the following challenges: a simple NLIDB system that works independently to the database domain neither the database structure. A system that supports more complex SQL queries (Joins, nested queries) helps the user types the query that is not necessarily long or simply the user can interact with the system through voice queries. Then, the system displays a suitable answer in a reduced time with appropriate error messages displayed in case of failure. Other new directions could be used in terms of increasing human-computer interaction by allowing our system to ask for clarification if the model cannot translate a given query.

References

1. Androutsopoulos, I., Ritchie, G. and Thanisch, P. (1995) « Natural language interfaces to databases: an introduction », Natural Language Engineering, Vol. 1, No. 1, pp.29–81.
2. N. Ranjan, K. Mundada, K. Phaltane, and S. Ahmad, "A Survey on Techniques in NLP," Int. J. Comput. Appl., vol. 134, no. 8, pp. 6–9, (2016).
3. Karam.A, Mustapha.M, Mourad.E, Brahim.E, Jilali.A, « Comparative study of existing approaches on the Task of Natural Language to Database Language », ICCSRE, p 1-6. (2019).
4. E. U. Reshma and P. C. Remya, "A review of different approaches in natural language interfaces to databases," in Proceedings of the International Conference on Intelligent Sustainable Systems, ICISS 2017, (2018).
5. Woods, William A, Ronald M Kaplan, and Bonnie Nash-Webber, « natural language information system », (1972).The lunar sciences.
6. Javubar SK, Jay A. (2015). «Natural language to SQL generation for semantic knowledge extraction in social web sources». Indian Journal of Science and Technology, 8(1): 1-10.
7. Mrs. Neelu Nihalani, Dr. Sanjay Silakari, Dr. Mahesh Motwani. "Natural language Interface to Database using Semantic Matching", International Journal of Computer Application, Vol. 31, no.11, Oct. (2011) ISSN: 0975 – 8887.
8. Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates, « Modern Natural Language Interfaces to Databases: Composing Statistical Parsing with Semantic Tractability », COLING (2004).
9. Harry R. TennantKenneth M. RossRichard M. Saenz, « Menu-based natural language understanding system », national computer conference and exposition (1984), Pages 629–635.
10. Hanane Bais, Mustapha Machkour, Lahcen Koutti, « An Arabic natural language interface for querying relational databases based on natural language processing and graph theory methods », International Journal of Reasoning-Based Intelligent Systems, (2018).
11. Mohit Dua, Sandeep Kumar, Zorawar Singh Virk, « Hindi Language Graphical User Interface to Database Management System », (2013) 12th International Conference on Machine Learning and Applications.
12. Bentamar Hemerelain, Hafida Belbachir, « Semantic Analysis of Natural Language Queries for an Object-Oriented Database », Software Engineering & Applications, (2010), 3, 1047-105.
13. XIAOFENG.M, S.WANG, and KAM FAI WONG, « Overview of A Chinese Natural Language Interface to Databases: NChiql », International Journal of Computer Processing of Languages, Vol. 14, No. 03, pp. 213-232 (2001).
14. Kailash Pati Mandala, Prasenjit Mukherjee, Baisakhi Chakraborty and Atanu Chattopadhyay. A novel Bengali Language Query Processing System (BLQPS) in medical domain. Intelligent Decision Technologies -1 (2019) 1–16.
15. Basik, F., Hättasch, B., Ilkhechi, A., Usta, A., Ramaswamy, S., Utama, P., & Cetintemel, U. «A Learned NL-Interface for Databases ».International Conference on Management of Data (pp. 1765-1768), (2018).
16. B Green, A Wolf, C.Chomsky, K Laughery, « BASEBALL: an automatic question answerer », western joint IRE-AIEE-ACM computer conference May (1961), Pages 219–224.
17. Woods, William A, Ronald M Kaplan, and Bonnie Nash-Webber, « natural language information system », (1972).The lunar sciences.
18. David L. Waltz, «An English language question answering system for a large relational database », Communications of the ACM, (1978).
19. G.Hendrix, « A NATURAL LANGUAGE INTERFACE », Computational Linguistics, vol.8, no.2, pp.55-61, (1982).
20. E.F.Codd, « A STEP Towards Realizing Codd's Vision of Rendezvous with the Casual User », Data

Base Management, Ed: North-Holland Publishers, (1974).

21. David H. D. Warren, Fernando C. N. Pereira, «An efficient easily adaptable system for interpreting natural language queries », Computational Linguistics (1982).
22. G. Ritchie, I. Androutsopoulos, P. Thanisch, « Masque/sql », (1993).
23. Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates, « Modern Natural Language Interfaces to Databases: Composing Statistical Parsing with Semantic Tractability », COLING (2004).
24. Yunyao Li, Huahai Yang, and H.V. Jagadish, Nalix: « an Interactive Natural Language Interface for Querying XML», SIGMOD (2005).
25. Hanane Bais, Mustapha Machkour, Lahcen Koutti, « Querying database using a universal natural language interface based on machine learning », (2016) International Conference on Information Technology for Organizations Development (IT4OD).
26. Hanane Bais, Mustapha Machkour, Lahcen Koutti, « An Arabic natural language interface for querying relational databases based on natural language processing and graph theory methods », International Journal of Reasoning-Based Intelligent Systems, (2018).
27. J. Gu, Z. Lu, H. Li, and V. O. K. Li. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. ArXiv e-prints, March (2016).
28. Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning.arXiv preprint arXiv: 1709.00103, (2017).
29. V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating struct queries from natural lang. using reinforcement learning," CoRR, vol. abs/1709.00103, (2017).
30. 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," in Proc. of the 2018 Conf. on Empirical Methods in Natural Lang. Process., pp. 3911–3921, (2018).
31. Jichuan Zeng, Xi Victoria Lin, Caiming Xiong, Richard Socher, Michael R. Lyu, Irwin King, Steven C.H. Hoi. Photon: A Robust Cross-Domain Text-to-SQL System. The 58th Annual Meeting of the Association for Computational Linguistics. ACL 2020.
32. IE-SQL: Text-to-SQL as Information Extraction. (2020) Association for Computing Machinery.
33. X. Zhang, F. Yin, G. Ma, B. Ge and W. Xiao, "M-SQL: Multi-Task Representation Learning for Single-Table Text2sql Generation," in IEEE Access, vol. 8, pp. 43156-43167, (2020), doi: 10.1109/ACCESS.2020.2977613.
34. Nathaniel Weir, Prasetya Utama, Alex Galakatos, Andrew Crotty, Amir Ilkhechi, Shekar Ramaswamy, Rohin Bhushan, Nadja Geisler, Benjamin Hättasch, Steffen Eger, Ugur Cetintemel, and Carsten Binnig. DBPal: A Fully Pluggable NL2SQL Training Pipeline. SIGMOD '20: Proceedings of the (2020) ACM SIGMOD International Conference on Management of Data, June 2020 Pages 2347–2361.
35. Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, Matthew Richardson. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7567–7578, July 5 - 10, (2020).
36. Amol Kelkar, Rohan Relan, Vaishali Bhardwaj, Saurabh Vaichal, Chandra Khatri, Peter Relan. Bertrand-DR: Improving Text-to-SQL using a Discriminative Re-ranker. WeCNLP (2020). Computation and Language (cs.CL).
37. DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, Dong Ryeol Shin. RYANSQL: Recursively Applying Sketch-based Slot Fillings for Complex Text-to-SQL in Cross-Domain Databases. (2020) Computation and Language (cs.CL).
38. Jaydeep Sen, Chuan Lei, Abdul Quamar, Fatma Ozcan2, Vasilis Efthymiou, Ayushi Dalmia, Greg Stager, Ashish Mittal, Diptikalyan Saha, and Karthik Sankaranarayanan. ATHENA++: natural language querying for complex nested SQL queries. (2020) proceedings of the VLDB Endowment, Vol. 13, No. 11, ISSN 2150-8097.
39. Ursin Brunner, Kurt Stockinger. ValueNet: A Neural Text-to-SQL Architecture Incorporating Values. (2020) proceedings of the VLDB Endowment.
40. Apostolos Glenis, Georgia Koutrika. NLonSpark: NL to SQL translation on top of Apache Spark. (2020) Association for Computing Machinery.