Journal of Theoretical and Applied Information Technology
30th June 2024. Vol.102. No. 12
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CHAT-SQL: NATURAL LANGUAGE TEXT TO SQL QUERIES BASED ON DEEP LEARNING TECHNIQUES

MAJHADI Khadija[1], MACHKOUR Mustapha[2]
[1,2]Team of Engineering of Information System Agadir, Morocco
E-mail: [1]khadija.majhadi@gmail.com, [2] m.machkour@uiz.ac.ma

ABSTRACT

The conversion of a text to a Structured Query Language (SQL) is a complex process that faces multiple challenges and a variety of problems. This is because the extraction of information stored in these databases requires the use of queries expressed in terms of a database query language, such as SQL. In addition, the Natural Language Interface to Database (NLIDB) is one of the most traditional applications of the NLP field that enables end-users to easily fetch data from databases. Recently, it has gained widespread attention, mainly because of the current success of Deep Learning techniques. The dominant NLIDB systems use the sequence-to-sequence approach. It is based on Long Short-Term Memory (LSTM) networks that include an encoder and a decoder method. In this article, we will tackle first the recent encoder/decoder approaches and analyze their pros and cons. Then, we will conduct an introductory summary of our suggested model for the NL to SQL problem. Namely, how this model can outperform the already existing solutions to enable it to manage the complex Natural Language questions-to-SQL generation queries in different contexts and cross-domain datasets. For this purpose, our work in this context will be focusing on facilitating access to the information stored in a database, by constructing a model that takes as input the natural language questions and translates them automatically to a structured language query. As a result, this model will offer a large number of database users simple and unlimited access to data with no need to learn any Database Query Language.

Keywords: *Natural Language Processing (NLP), Machine Learning (ML), Deep Learning (DL), Natural Language Interface to Database (NLIDB), Long Short-Term Memory (LSTM), Structured Query Language (SQL).*

1 INTRODUCTION

Natural Language Processing (NLP) is one of the most challenging areas of research in Artificial Intelligence (AI) that focuses on designing and developing computer systems that can analyze, understand, and synthesize natural human languages [1]. It is used in Human-Computer interaction for information retrieval, machine translation, and linguistic analysis. Today, NLP is present in many everyday applications and more particularly in the area of information extraction such as database querying.

Nowadays, databases represent an essential source of information for all its users, experts or non-experts. However, to get access to this information, it is necessary to use a query language like Structured Query language (SQL). In this context, the end-users who do not know the structure of a database or the syntax used to interact with it will not be able to extract data from these databases. One solution to this problem is to provide an interface to query the database in Natural Language that can take questions or queries from users in natural language, transform it to a formal language like SQL, then execute the generated SQL query, and show the desired result to the end-users [2]. Natural Language Interface to SQL is an appealing area of research that appeared in the late 1960s and early 1970s [3]. It is an intermediate layer between the Relational Database Management System (RDBMS) and the end-user that overcomes the communication gap between them. The NLIDB system is an intelligent and flexible tool since it is simple to use, and allows the users to use NL sentences to retrieve information from a database without using any intermediate language. Also in that context, we have Text-to-SQL which is the task of generating SQL queries from natural language questions

In the rest of this article, section 2 represents the literature review. Section 3 describes the challenges of natural language text to SQL problem. Section 4 illustrates different approaches and datasets that have been used for translating natural language questions to corresponding SQL queries. Section 5 shows the overview and the simulation results of our proposed approach. Then, an evaluation is presented for the approval of the contribution. Finally, the conclusion provides a summary of the work done and some perspectives arising from this study.

2 RELATED WORKS

Numerous research efforts have been dedicated to the field of Natural Language Processing (NLP), and one of its significant achievements is the development of Natural Language Interface to Database (NLIDB) systems. The success observed in this domain is attributed to both the practical benefits derived from NLIDB systems and the effective functioning of NLP within single-database domains. Generally, databases often operate within sufficiently limited domains, enabling the successful resolution of ambiguity issues in Natural Language (NL).

The earliest research initiatives in this area date back to the 1960s, with the creation of several systems since that time. Notably, BASEBALL and LUNAR, introduced in the late sixties, marked the inception of operational NLIDB systems. The BASEBALL system was specifically designed to address inquiries related to baseball games, while LUNAR focused on the chemical analysis of moon rocks, utilizing an Augmented Transition Network. However, these systems were non-reconfigurable, tailored for specific domains, and lacked ease of modification for interfacing with different databases.

In the late 1970s, various research prototypes were developed, including LIFER/LADDER (1978), designed to provide information about US Navy ships and acknowledged as one of the initial effective NLIDB systems. Employing semantic grammar techniques encompassing both syntactic and semantic processing, LIFER/LADDER represented a significant advancement. However, systems reliant on semantic grammar faced challenges when applied to different application domains, requiring the development of a distinct grammar for each domain where LADDER was employed. Another noteworthy system from the 1980s is CHAT-80 (1980), recognized as one of the most referenced Natural Language Processing (NLP) systems of that era. Implemented in Prolog, CHAT-80 involved the conversion of English queries into Prolog expressions, which were then assessed against the Prolog database. The code for CHAT-80 was widely disseminated and served as the basis for several experimental systems, including MASQUE (Modular Answering System for Queries). A modified version of MASQUE, known as MASQUE/SQL, was developed. This system translates the NL query into an intermediate logic representation and subsequently translates the logic query into SQL. However, it has some shortcomings; in cases of SQL query failure, the system does not pinpoint the specific part of the query responsible for the failure. Additionally, it is domain-dependent and requires configuration for use in other knowledge domains.

In recent times, several Natural Language interface to database systems have been developed, including the PRECISE system (2004) and NALIX (Natural Language Interface for an XML Database) (2006). The PRECISE system, originating from the University of Washington, stands out as an exemplary instance of approaches pertinent to the design of NLDBIs that are independent of specific databases. Through the integration of state-of-the-art statistical parsers and a novel concept of semantic tractability, PRECISE emerges as a highly reconfigurable system, demonstrating impressive performance in handling semantically tractable queries. However, the system encounters challenges in dealing with nested structures. On the other hand, NALIX, developed at the University of Michigan, represents the first generic interactive natural language query interface to an XML database (extensible markup language). NALIX employs a three-phase process for transformation: generating a Parse tree, validating the parse tree, and translating the parse tree into an XQuery expression. Consequently, NALIX can be categorized as a syntax-based system. The system utilizes Schema-Free XQuery as its database query language, offering the advantage of not requiring an exact mapping of a query to the database schema, as it can automatically identify all relevant relations based on specific keywords.

The previous models address the challenge as a semantic parsing task in natural language processing (NLP) for translating text to SQL, utilizing illustrated datasets. Some of these models incorporate linguistic techniques or heuristics to improve the quality of generated queries or reduce the output space.

Seq2sql [13] was one of the early works introducing a sequence-to-sequence structure for translating text to SQL using reinforcement learning [4]. It achieved a 60% execution accuracy on the WIKISQL dataset. SQLNet, another model, employs a column attention mechanism to highlight relevant parts of user questions concerning database columns, using the WIKISQL dataset. This model consists of several sub-modules, each responsible for predicting a single token, not a sequence.

PHOTON [5] is a robust cross-domain text-to-SQL system comprising a powerful neural semantic parser, a human-in-the-loop question corrector suggesting possible corrections, an SQL query executor, and a natural language response generator. The Information-Extraction-to-SQL (IE-SQL) [6] approach involves two models: a BERT-based extraction model to identify slots in the input sentence and a BERT-based linker mapping columns to the table schema for generating executable SQL queries. M-SQL [7], utilizing pre-trained BERT, consists of an encoder for Chinese word vector representation, column representation, and eight sub-models predicting the select and where clauses of the SQL statement.

RAT-SQL [8] relies on relation-aware schema encoding and linking for text-to-SQL parsers, incorporating a self-attention mechanism for schema encoding, linking, and feature representation. SQLNet adopts a sketch-filling approach, focusing the model on essential parts of the query. RYANSQL (Recursively Yielding Annotation Network for SQL) [9] is a neural network approach applying sketch-based slot fillings for complex text-to-SQL in cross-domain databases.

ATHENA++ [10] exemplifies natural language querying for complex nested SQL queries, combining linguistic patterns from NL queries with deep domain reasoning, using ontologies to capture domain schema semantics on the FIBEN benchmark dataset. The system architecture includes a Translation Index, Domain Ontology, Ontology to Database Mapping, and Query Translator.

ValueNet [11] is an end-to-end text-to-SQL system, aiming to use all available information from the base data as input for the neural network architecture. This new architecture extracts values from user questions and predicts possible value candidates not explicitly mentioned in the question.

Finally, an extensive prior study of various approaches for the Text-To-SQL domain provided us with an obvious roadmap for this research [21]. Therefore, a trial to find an efficient solution for converting a text to SQL, while implementing a model for language representation that will only need the use of a human Natural Language to access databases easily, saving time and effort.

3 CHALLENGES OF NATURAL LANGUAGE TEXT TO SQL PROBLEM

The process of translating an NLQ to SQL involves addressing challenges related to both understanding the input NL query and constructing the correct SQL query, ensuring syntactic and semantic accuracy, based on the underlying database schema. Figure 1 provides an example of a text-to-SQL transformation. The objective is to generate an equivalent SQL query that maintains the intended meaning. This SQL query should be valid for the specified DB and, upon execution, yield results aligned with the user's intent.

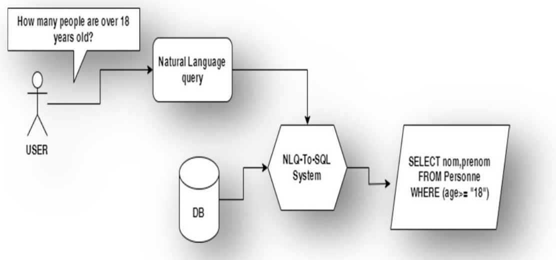
How many people are over 18 years old?
USER
Natural Language query
DB
NLQ-To-SQL System
SELECT nom,prenom FROM Personne WHERE (age>= "18")

Figure 1:Example of Natural Language Text-To-SQL

3.1 Natural Language challenges

Ambiguity is a fundamental characteristic of natural language, inherently permitting the creation of expressions open to multiple interpretations. Various types of ambiguity exist, and the most prevalent ones are outlined below:

Lexical ambiguity, also known as polysemy, occurs when a single word possesses multiple meanings. For instance, the word "Paris" could refer to a city or an individual.

Syntactic ambiguity arises when a sentence can be interpreted in multiple ways based on its syntactic structure. For example, the question "Find all France movie directors" could be parsed as either "directors who have directed France movies" or "directors from France who have directed any movie."

Semantic ambiguity occurs when a sentence has multiple interpretations at the semantic level.

Context-dependent ambiguity involves a term having varying meanings depending on factors such as the query context, the data domain, and the user's objectives. Notably, terms like "top" and "best" exemplify this phenomenon. Consider the query context: for the question "Who was the best runner of the marathon?" within the context of completing the race faster (min operation), the answer should reflect the speed.

Examining the domain, in a movie database, the query "Return the top movie" might imply ranking based on the number of collected ratings. Conversely, in a football database, for the query "Return the top scorer," the term "top" is associated with the number of goals scored. Moreover,

depending on the user, the query "Return the top product" would yield different results. For a business analyst, it should return the most profitable products, whereas for a consumer, it should return the top-rated products.
In natural language, two sentences can convey the same meaning while being phrased in entirely different ways. For example, the sentences "How many people live in Moroco?" and "What is the population of Moroco?" have equivalent meanings and can be translated into the same SQL query. However, the second sentence may be more straightforward for a system because it is likely that a "population" attribute exists in the database schema, allowing for a higher confidence in inferring the user's intent. Paraphrasing also involves synonymy, where multiple words, such as "movies" and "films," share the same meaning.
Inference involves the recognition that a query may lack essential information for a system to fully comprehend it. The system must deduce the missing details based on the provided context. Elliptical queries are sentences where one or more words are omitted, yet they remain understandable within the sentence's context. Follow-up questions are a common aspect of human conversations. After asking a question and receiving an answer, a follow-up question is posed with the assumption that the context of the initial question is understood. For instance, "Q: Which is the capital of Morocco? A: Rabat. Q: What about Indounisia?" Without the first question, the second one may seem nonsensical, but within the query context, it becomes evident that it is inquiring about the capital city of Morocco.
User mistakes, such as spelling errors or syntactical and grammatical errors, further complicate the translation problem.

3.2 SQL challenges

SQL syntax is characterized by its strict rules, resulting in limited expressivity compared to natural language. Certain queries are straightforward to articulate in natural language but may translate into complex SQL queries. Additionally, while a sentence in natural language may be comprehensible despite containing some mistakes, SQL is less forgiving. An SQL query derived from a natural language query must be both syntactically and semantically correct to be executable over the underlying data [15,17].
Database Structure: The user's conceptual model of the data, including entities, their attributes, and relationships described in the data, may not align with the database schema, leading to various challenges. The vocabulary gap represents disparities between the terms employed by the user and those utilized in the database [18].
Schema ambiguity arises when a portion of the query may correspond to more than one element in the database. Implicit join operations occur when segments of a query are translated into joins across multiple relations. Also, entity modeling presents the challenge where a set of entities may be modeled differently, such as distinct tables or rows (or values) within a single table.

4 ANALYSIS

4.1 Existing datasets

To develop a neural text-to-SQL system, it's crucial to take into account the datasets available for training and evaluation. Additionally, the evaluation methodology plays a significant role in testing and comparing the system's performance to other models. A text-to-SQL dataset, or benchmark, encompasses a collection of natural language (NL) and SQL query pairs defined over one or more databases. Various datasets have been released for either training or evaluating models that translate natural language questions into corresponding SQL queries. These datasets exhibit differences in terms of the number and types of queries they include. Here are some of the most commonly utilized ones:
WikiSQL: The most widely used and extensive benchmark dataset, WikiSQL is a mono table dataset comprising 26,531 tables and 80,654 pairs associated with a given single table. It does not support joins and nested queries, making it suitable for evaluating simple models with only one column in the SELECT clause and one table in the FROM clause. Tables are extracted from HTML tables on Wikipedia. Each SQL query is automatically generated for a given table under the constraint that it produces a non-empty result set. NL queries are generated using templates and paraphrased through Amazon Mechanical Turk.
ATIS: [14] Primarily used for semantic parsing, ATIS focuses on flight booking and includes a database of 25 tables and 5,410 query pairs. Most questions can be answered with a single relational query, but the dataset lacks grouping or ordering queries.
GeoQuery: Comprising seven tables in the US geography database, GeoQuery consists of 880 query pairs, including 256 nested SQL queries. Unlike WikiSQL, all queries in ATIS and GeoQuery are specific to a single domain, with both benchmark databases featuring various

queries, including join and nested queries.

MAS (Microsoft Academic Search): Similar to ATIS and GeoQuery, MAS focuses on social academics. It comprises a database of 17 tables and 196 query pairs, featuring various SQL queries that include join, grouping, and nested queries.

Spider [16] is a large-scale cross-domain NL2SQL benchmark dataset with 200 databases spanning 138 different domains and 10,181 query pairs distributed across training, development, and test sets. Spider addresses perceived limitations in existing benchmarks, offering a more diverse range of queries, including join and nested queries.

We can classify these datasets into two types: one that comprises queries with limited structures, like the WikiSQL dataset, and SQL databases like GEOQuery, MAS (academia), YELP, ATIS, and Spider. These databases are publicly available and contain real data accumulated over the years. Recent studies often evaluate their models on the Spider dataset, which is a compilation of various databases such as restaurants, geography, and academia. However, only 10% of the pairs in the training set originate from the aforementioned databases (approximately 1,659 queries). The remaining queries (about 7,000) are from unknown databases, defined by the creators of Spider, diminishing its credibility as a dataset for the text-to-SQL task. With just 10,181 question/SQL query pairs in the development, training, and test sets combined, Spider is insufficient for assessing the integrity, reliability, and relevance of any model. Despite Spider offering a diversified range of query difficulties, including easy, medium, hard, and extra hard, encompassing nested queries, the presence of the GROUP BY command, as well as keywords like UNION and EXCEPT, the limited number of queries poses a challenge in accurately evaluating a model. Additionally, the dataset is unbalanced, with only 8.2% of samples containing nested queries and a mere 3.8% including HAVING/GROUP BY commands.

4.2 Existing approaches

Syntactic parsing: Syntactic parsing represents a category of solutions that relies on syntactic linguistic techniques. In this class of solutions, a predominant approach is the utilization of guided (strict) grammar for user input. Models based on strict grammar typically necessitate users to adhere to a specific word order when entering a sentence into the system; otherwise, the sentence may go unrecognized and consequently not be translated. Alternatively, there is the option of employing free grammar, which enhances the models' flexibility to various sentence structures but can impact the quality of the outputs. These solutions are often deemed less intelligent, requiring extensive hand-engineering techniques to deduce the relevant parts in the input sentence for accurate conversion to SQL.

The syntactic parsing approaches have in general some common steps. They start by defining the grammatical categories of the input sentence to identify the nature of each word. This means the extraction of the part of speech as shown in the example bellow:

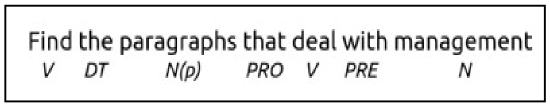

Figure 2: Part of speech tagging

In this example, the tokens within the sentence "find the paragraphs that deal with management" consist of a verb, an article (a determiner), a noun, a pronoun, a verb, a preposition, and finally, a noun. This breakdown is crucial for models that rely on syntactic parsing, as these tags play a vital role in constructing the syntax tree, facilitating the identification of dependencies among words in the question. The application of context-free grammar allows for the creation of the syntax tree. It's worth noting that these rules are not specifically tailored to the Text-to-SQL task and can be employed for other syntactic parsing purposes. Figure 3 illustrates the established tree resulting from the part-of-speech tagging step.

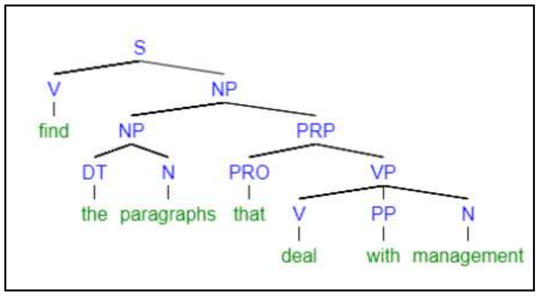

Figure 3: Syntax tree of the sentence in the example

The tree undergoes additional processing to be transformed into an intermediate representation. Various treatments may be applied in this step, such as employing morphological analysis to gather valuable information about each token or transforming the tree into a transitional representation to bring the initial question closer to the target query. Numerous intermediate forms have been utilized in the studies under examination, with XML being a notable example.

Semantic parsing: Many models rely on deep learning as a primary method for predicting the correct elements in the target query. Often adopting a free grammar style, these models are regarded as end-to-end solutions that eliminate the need for manual engineering. In the majority of cases, these models only require an annotated dataset for training. They can be directly trained on a corpus of pairs (Question/Target Query) without the need for interactivity or feedback messages from the system. This is accomplished by employing a sequence-to-sequence decoding approach to generate SQL queries.
For translating a source sentence into a target sentence, the sequence-to-sequence model (Seq2Seq) [12] uses an approach that consists of an encoder and a decoder, which is implemented by an RNN or an LSTM [19]. The encoder takes a source sentence (input data) and reads it via a fixed-size context vector, while the decoder takes the context vector C and generates a target sentence. Recurrent neural networks (RNNs) use an input sequence of vectors [x1x2 . . . xτ] of length τ as well as an initially hidden state h0 and generate a sequence of hidden states [h1,h2 . . . hτ]. The sequence of output vectors [y1,y2 . . . yτ]. Specifically, ht at time step t is calculated by:

$$ht = f\ theta\ (xt, ht-1) \qquad (1)$$

Where f theta is a function with a parameter θ, that is generally referred to as an RNN cell.
If an RNN cell is implemented as just a fully connected layer with an activation function, it will not efficiently accumulate information from previous time steps in its hidden state of the RNN. Such a basic RNN would not effectively handle long sequences and face the notorious vanishing and exploding gradient problem. To avoid and namely solve the problem, it is suggested to function the long short-term memory (LSTM), gated recurrent units (GRUs), or residual networks (ResNets). For Instance, an LSTM cell maintains an additional cell state ct that saves information over time and three gates so that it can regulate the flow of information into the cell or out of it. That is to say, ht and ct are computed using the gates from ct−1, ht−1, and xt.
The general process in the Seq2seq approach is made by breaking down the input sentence each time into tokens that are used to generate the SQL query. Each token concerns a part of the SQL syntax: SELECT, AGGREGATION, From, where, etc. The output token from the previous LSTM layer is fed as the input token to the next layer operation until the token <END> is generated as shown in figure.4.

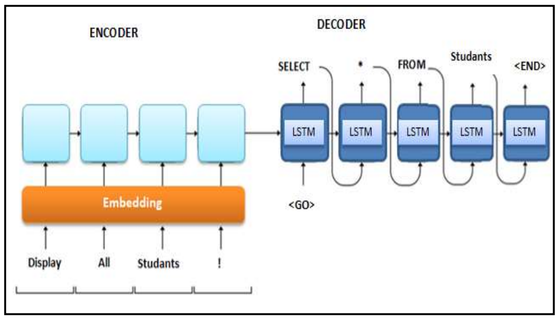

Figure 4: Seq2seq General process

4.3 Evaluation

The first existing NLIDB models that are based on the use of shallow and end-to-end approaches have shown very limited and low-quality results and the majority of them fail when they are tested with new datasets or new schemas that haven't been seen in the training and the development collections. But, the most current NLIDB systems that used Deep Learning techniques [20] to translate Natural Language queries to SQL achieved encouraging results on the challenging Spider benchmark dataset-based encoder-decoder architecture as shown in Table1.

Table 1: Execution accuracy of recently developed systems on the challenging Spider benchmark dataset based encoder-Decoder architecture.

Model	Dataset	Accuracy (%)	
		Dev	***Test***
RAT-SQL	**Spider**	**62,7**	**57,2**
Bertrand-DR	**Spider**	**57,9**	**54,6**
M-SQL	**TableQA**	**91,86**	**92,1 3**
Photon	**Spider**	**63,2**	
IE-SQL	**WikiSQL**	**94,2**	
Athena++	**Spider**	**78,82**	

5. PROPOSED MODEL

5.1 Overview

We suggest an approach based on two phases: The first one consists of ensuring the pre-processing of Natural Language input, while the second one is devoted to the translation of this question into a structured format, namely the SQL query. Our tool allows users to easily access the data using natural language as a means of communication. In this study, we are interested in data collected from the SPIDER database (introduced in section 3 part 1). The pre-processing phase is triggered once a user enters a question. The system exploits Deep Learning techniques once the question is processed, to translate this question into a SQL query.

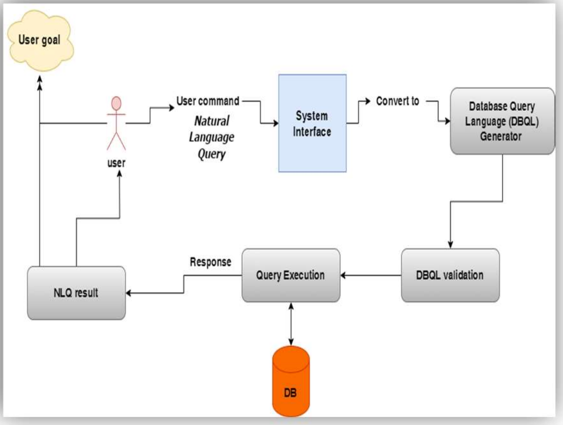

Figure 5: CHAT-SQL General process

Natural language understanding The NLU unit's responsibility lies in converting user utterances into a predetermined semantic framework based on the system's conventions. This involves tasks such as slot filling and intent detection, where the aim is to render the input understandable for the system. For instance, the intent could range from a simple greeting like "Hello" or "Hi" to an informative statement such as "I like Indian food," where the user provides additional details. The slots, which can vary widely based on the context, might include entities like actor names, prices, start times, or destination cities.

This interplay between intents and slots underscores the closed-domain nature of the Chatbot. The process of slot filling and intent detection is often framed as a sequence tagging problem. Hence, the NLU component is commonly built using an LSTM-based recurrent neural network augmented with a Conditional Random Field (CRF) layer. One prevalent model in this domain employs a sequence-to-sequence architecture leveraging bidirectional LSTM networks to simultaneously fill slots and predict intent. Conversely, an attention-based RNN accomplishes the same task with a different architectural approach.

Natural Language Generation (NLG) involves the creation of text from a given meaning representation, essentially serving as the inverse of natural language understanding. NLG systems play a crucial role in tasks like text summarization, machine translation, and dialog systems. In NLG, the system formulates a response in the form of a semantic frame, which is then translated into a natural language sentence understandable to the end user.

NLG components can take the form of rule-based or model-based systems, and sometimes a hybrid of both. Rule-based NLG generates predefined template sentences based on a given semantic frame, but they are often limited in their adaptability and lack generalization power. While general-purpose rule-based generation systems exist, they can be challenging to tailor to specific, task-oriented applications due to their broad applicability. In contrast, machine learning-based NLG systems, which are more prevalent in modern dialog systems, leverage various input sources such as a content plan representing the intended message, a knowledge base providing domain-specific entities, a user model imposing constraints on output utterances, dialog history to avoid repetition, and referring expressions, among others. These trainable NLG systems offer greater flexibility and adaptability compared to rule-based approaches. Our System CHAT-SQL provides users with the ability to ask questions directly in the chat interface with the system in writing or verbally; as our system can interpret and translate the audio voices into text and then respond to the requests submitted. Once the question is valid, the system provides the answers in the form of a written or spoken SQL query as shown in figure 5. It is important to mention that our system responds to different types of simple or composed queries.

Interactivity hasn't been adopted deeply in the previously proposed systems; however, the interactivity between the user and the system can take different forms, including initial questions to the user, feedback questions, error corrections, etc. One of the solutions is asking a sequence of questions to better understand the need and the wanted query. This might be at different stages and it can be mixed with other techniques. The system can ask for example about the column's name to be included in the 'Select' clause or maybe the tables to include in the 'From' one. It is a new direction present in the feedback messages showing the

corrections and suggestions to guide and limit the space of outputs to predict the right and correspond answer to their requests.

5.2 Evaluation

To evaluate the performance of our system in translating a natural language question into a structured query, namely a SQL query, we propose to use two well-known measures: recall and precision. We propose to evaluate whether the generated SQL queries are an accurate representation of the proposed question. Therefore, we provided several test questions to the system after the training. When a user enters a question, the system can provide either correct answers, or queries that provide incorrect answers due to a mistranslation of the question into SQL, or the system cannot fully ensure the translation of the question into an SQL query. Our system considers different types of queries even the most complex ones. In this context, the two measures of recall and precision can be defined as follows:

The recall can be described as the number of queries providing correct answers, related to the number of generated SQL queries.

The precision represents the number of generated queries providing correct answers, relative to the total number of suggested questions.

In the same context, we adopted an additional measure used to evaluate the effectiveness of the system called Accuracy. The accuracy focuses on the predictive capability of a model in the set of experimental samples.

Based on these measures, a set of test questions formulated in natural language (350 questions) was proposed. The generated queries are checked, and then we find different types of answers are returned when executing these queries. The results are given in the following table:

.

Table 2: Execution accuracy of recently developed systems on the challenging Spider benchmark dataset based encoder-Decoder architecture

Total number of test questions	**350**
Number of queries providing correct answers	**290**
Number of queries providing incorrect answers	**60**
Number of queries generated by our system	**330**
Precision	**82,8%**
Recall	**87,8%**
Accuracy	**89%**

From the recall (87,8%) and precision (82,8%) rates obtained, we can see that our system has achieved satisfactory results. Thus, the Accuracy rate obtained is about 89%, which confirms the performance and efficiency of our system.

6. CONCLUSION AND FUTUTRE WORK

This research paper explores a comprehensive study of a generic natural language query interface for a database, employing Deep Learning techniques to handle lengthy and complex SQL queries. Natural Language Interface to Databases (NLIDB) represents a dynamic field in automatic language processing, aiming to comprehend requests articulated in natural languages commonly used by non-technical users and generate corresponding responses. Functioning as a human-machine interface, our system facilitates user query input through a dictionary or voice interaction. The system promptly provides suitable answers, accompanied by relevant error messages in case of failure.

Experimental results indicate the system's satisfactory performance, delivering reasonable and accurate responses across various types of natural language queries, including those in different languages, queries involving joins, complex structures, and lengthy queries. The achieved accuracy rate stands at approximately 89%, affirming the effectiveness and efficiency of our system. An innovative aspect of our approach involves enhancing human-computer interaction by allowing the system to seek clarification when faced with queries it cannot translate accurately. Looking ahead, our future work aims to further develop a mobile application, ensuring simplicity for users of all types. With a single click, users can receive direct answers to their queries on their smartphones [22].

REFERENCES:

[1] Author [1] GMay 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).

[2] 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).

[3] K.Majhadi and M.Mustapha. The history and recent advances of Natural Language Interfaces for Databases Querying, E3S Web of Conferences 229, 01039 (2021), The 3rd International Conference of Computer Science and Renewable Energies.

[4] Victor Zhong, CaimingXiong, Richard Socher, Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning (2017).

[5] JichuanZeng, Xi Victoria Lin, CaimingXiong, 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, 2020.

[6] IE-SQL: Text-to-SQL as Information Extraction. (2020) Association for Computing Machinery.

[7] 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.

[8] Bailin Wang, Richard Shin, Xiaodong Liu, OleksandrPolozov, 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).

[9] DongHyun Choi, MyeongCheol 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).

[10] JaydeepSen, Chuan Lei, Abdul Quamar, Fatma Ozcan2, VasilisEfthymiou, AyushiDalmia, Greg Stager, Ashish Mittal, DiptikalyanSaha, and KarthikSankaranarayanan. ATHENA++: natural language querying for complex nested SQL queries. (2020) proceedings of the VLDB Endowment, Vol. 13, No. 11, ISSN 2150-8097.

[11] Ursin Brunner, Kurt Stockinger. ValueNet: A Neural Text-to-SQL Architecture Incorporating Values. (2020) proceedings of the VLDB Endowment.

[12] K. Ahkouk, M. Machkour, K. Majhadi, R. Mama, SEQ2SEQ VS SKETCH FILLING STRUCTURE FOR NATURAL LANGUAGE TO SQL TRANSLATION. 5th International Conference on Smart City Applications, 7–8 October 2020.

[13] V. Zhong, C. Xiong, and R. Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. CoRR, abs/1709.00103, 2017.

[14] P. J. Price. Evaluation of spoken language systems: the ATIS domain. In DARPA Speech and Natural Language Workshop, pages 91–95, 1990.

[15] J. M. Zelle and R. J. Mooney. Learning to parse database queries using inductive logic programming. In AAAI, pages 1050–1055, 1996.

[16] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. R. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In EMNLP, pages 3911–3921, 2018.

[17] Hyeonji Kim, Byeong-Hoon So, Wook-Shin Han, Hongrae Lee. Natural language to SQL: Where are we today? Proceedings of the VLDB Endowment, Vol. 13, No. 10 ISSN 2150-8097.

[18] 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).

[19] X. Zhang, M. H. Chen and Y. Qin, "NLP-QA Framework Based on LSTM-RNN, 2nd International Conference on Data Science and Business Analytics (ICDSBA), 2018, pp. 307-311, doi: 10.1109/ICDSBA.2018.00065.

[20] Abbas, S., Khan, M.U., Lee, S.U.-J., Abbas, A., Bashir, A.K.: A review of nlidb with deep learning: findings, challenges and open issues. IEEE Access. 10, 14927–14945 (2022).

[21] Deng, N., Chen, Y., Zhang, Y.: Recent advances in text-to-SQL: a survey of what we have and what we expect. In: Proceedings of the 29th International Conference on Computational Linguistics, pp. 2166–2187. International Committee on Computational

Linguistics, Gyeongju, Republic of Korea (2022).

[22] P. Parikh et al., "Auto-Query - A simple natural language to SQL query generator for an e-learning platform," IEEE Global Engineering Education Conference (EDUCON), Tunis, Tunisia, 2022, pp. 936-940, doi: 10.1109/EDUCON52537.2022.9766617, (2022).