Journal of Theoretical and Applied Information Technology
30th November 2020. Vol.98. No 22

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

PIPELINE AND DEEP LEARNING APPROACH FOR NLIDB: A COMPARATIVE STUDY

[1]SHANZA ABBAS, [2]MUHAMMAD UMAIR KHAN, [3]SCOTT UK-JIN LEE*, [4]ASAD ABBAS
[1,2] Department of Computer Science and Engineering, Hanyang University, Republic of Korea
[3]Department of Computer Science and Engineering, Major in Bio Artificial Intelligence, Hanyang University, Republic of Korea
[4]Faculty of Information Technology, University of Central Punjab, Lahore, Pakistan

*Corresponding Author: [3]SCOTT UK-JIN LEE

E-mail: [1]shanza92@hanyang.ac.kr, [2]mumairkhan@hanyang.ac.kr, [3*]scottlee@hanyang.ac.kr, [4]asadabbas.grw@ucp.edu.pk

ABSTRACT

Databases are integral part of current world's scenario of rich technology. Greater amount of the data in the world is stored in the databases. That amount of data storages can be utilized for various purposes in data science world. Besides potential usage and benefits of available data amounts, the requirement of formal language to access the databases is a huge hurdle. Structured Query language (SQL) is one of such formal languages to access the database. Besides its impact and powerful as a language it is not a common knowledge. Therefore, domain experts of some particular databases cannot access their data freely and easily. Web interfaces to access that data has their own limitation and do not fulfil the purpose to the maximum of the potential of data. Natural Language Interface to Database (NLIDB) system is natural solution for such problems. Text to SQL task in NLIDB system is being experimented with since 70s. Previously it was based on integrated methods and techniques from Natural Language Processing (NLP) and Data Science areas, those integrated frameworks generally known as pipeline methods. Recently, machine learning showed promising performance for the solutions to semantic problems. Which is why, deep learning had been adopted for text to SQL task as well. Currently NLIDB systems research is going on with both of the approaches of pipeline methods and deep learning methods in parallel. It is important at this time to analyze the latest work with both approaches and compare and identify their unique challenges and issues as well as findings and potential of both approaches for the NLIDB systems. In this paper, a comparative analysis is presented to find out the achievements and issues of NLIDB with pipeline methods and with deep learning methods regarding each of them.

Keywords: *Structured Query language, Natural Language Processing, Natural Language Interface to Databas*

1. INTRODUCTION

Currently majority of the data in the world is stored in the form of databases. That data is being used for various purposes including huge contribution in the research. A prominent hurdle from that data being utilized to the full of its potential is the need of structure query languages to access the databases. Structured query languages are not possible for everyone to learn [1]. This problem creates a gap between the domain experts and the database experts who can access the database. Text to SQL task propose the possible solution for this problem in the form of NLIDB [2]. Natural Language Interface to Database (NLIDB) provides and easy front end for users to access the data from databases without using the structured query language.

Building natural language interfaces to databases (NLIDB) is a long-standing open problem and has significant implications for many application domains [3]. Besides many advantages NLIDB is not adopted widely and considered traditionally difficult area, because of inherently ambiguous nature of natural language. Most challenging part of whole process is understanding user's intentions, because of inherently ambiguous nature of natural language [4]. In Pipeline method this part is called keyword mapping. This part has been focused significantly in recent efforts for the work area. Despite of introducing various upgradations for this

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

part, it is still not resolved effectively. Hence, NLIDB system overall is still unable to get practical enough accuracy results. Recently the area

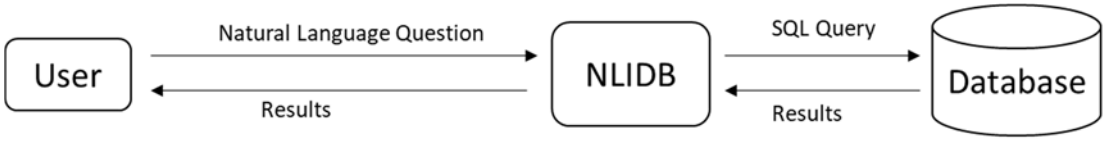

Figure 1. Illustration of NLIDB system.

of machine learning has been raised and came

in limelight significantly [5]. It opened gates for many other areas to boom again. Semantic parsing is one of such example areas as shown in figure 1. Semantic parsing received a new interest from researchers after machine learning ideas. With recent advancements in the field of deep learning and semantic parsing, generating SQL queries from natural language questions has gained a renewed interest as well [6-9]. Therefore, NLIDB with deep learning is an arising focus of this area. Deep learning methods integrating Natural language processing techniques along with them and showed potential results. Although initial accuracy results are less than pipeline methods still the learning rate of deep learning model is promising for possible improved NLIDB systems [10]. Also, NLIDB with deep learning showed potential solution for the critical hurdles faced by pipeline methods for the task. Although deep learning methods for NLIDB looks potential solution for existing limitations in Pipeline methods, but still they are also not being able to produce any realistically implementable results [11]. Currently, both approaches, NLIDB with pipeline methods and deep learning methods are being studied and tested in parallel. Much work has been invested in NLIDB with deep learning in last few years as well as pipeline methods has also not being dropped. Various novel ideas for the problems and challenges has been proposed bringing up new issues linked with them in the light. Similarly pipeline methods also known as grammar based, lexical based systems, have proposed new ideas with latest issues and problems[12]. With ongoing work based on both approaches it is need of time to analyze the recent efforts and compare the issues and benefits from both approaches. It is also important to find out the limitations and hurdles for each of them. A comparison of both approaches will be useful to find out the brief picture of situation in the rea. Therefore, we aim to display a brief comparative analysis for both approaches in this paper. Our goal

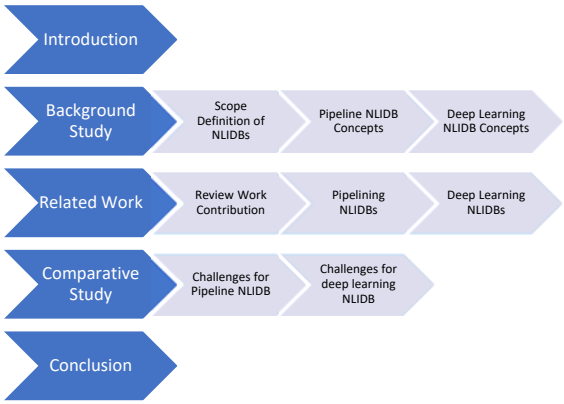

Figure 2. Taxonomy of the paper.

is to put together their benefits and limitations along with available solutions. This will provide an overview about current research trends and achieved performance for each of these architectural approaches as shown in figure 2.
Remaining paper is organized as Background study in section 2, Related Work in section 3, Comparative Study in section 4 and Conclusion is drawn in section 5.

2. BACK GROUND STUDY

Natural Language Interface to database has various basic concepts like keyword mapping, sentence analysis, SQL generation for pipeline methods. There are some basic concepts about deep learning NLIDB concepts as well like encoder, decoder, word embedding, sequence to sequence learning etc. All these basic concepts for both of the approaches are important to explain for better understanding of the pros and cons for both approaches. Following are these ideas and concepts explained in detail one by one. Scope of an NLIDB system is common concept for both type of

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

NLIDBs. It is explained further below.

2.1. SCOPE OF EXISTING NLIDB SYSTEMS

It is important to know until now what kind of natural language queries are supported in existing NLIDB systems. That can be characterized in 4 types of systems described below.

A- Ad-hoc Natural Language Queries:

In an ideal situation, NLIDB systems should be able to work with ad hoc style of natural language questions. There are some NLIDB systems such NLIR which are working toward this idea [13]. But overall NLIDB systems are not able to parse ad hoc queries and this is an open issue for the area. In order to handle the ad hoc queries NLIDB systems need to rely on parser error handling heavily. Generally controlled natural language questions are the pattern to avoid parsing errors. Controlled natural language questions meaning restricted type of question support in the system. Such as [14] suggested the tractability of queries to find the subset of the natural language questions for the purpose of translating into structured queries. [15] defined a grammar beforehand, based on which natural language questions could be formatted for the convenience of the system. Similarly, [16] also had a defined criteria foe natural language questions to limit scope for minimizing parsing errors. Contribution by [17] suggested the domain specific template based natural language question structure. With such systems parsing errors are minimized in NLIDB area but it brought other challenges in the light such as:

1- Training of users to understand and learn the structured language rules enough to use it effectively and in error free manner.
2- Making sure that user can express the complete requirement in the question while remaining under the scope of structure rules.

Another factor that defines the scope of NLIDB system is whether it is a conversational system or not. Impact of this factor is important to understand for NLIDB systems overall. Therefore, detailed explanation is provided to understand this idea completely.

B- Stateful NLIDB System:

NLIDB systems that keep the context and able to answer the follow up questions are conversational of stateful NLIDB systems [18]. Ideally NLIDB system should be conversational and should have the capability to handle follow up queries based on the history of conversation. Significant contributions in such regard has been made by [18] and [19].

C- Non Conversational NLIDB System:

NLIDB system that handle the queries separately and independently from each other with no context with the history queries, they are stateless or non-conversational NLIDB systems. Current work in the area of text to SQL task most of the work is being done in stateless NLIDB systems [20]. Interaction history might be utilized only to add the quality in current query prediction but not utilized to provide complete conversational context. Sometimes, the pattern of a user interaction might appear as conversational with a stateless NLIDB system as well [21].

Table 1. NLIDB system scope categories

Categories of questions	Description
Ad-hoc Natural Language Queries	Unstructured spontaneous natural language questions.
Stateful	Conversational NLIDB system
Stateless	Non-Conversation NLIDB system

2.2. PIPELINE NLIDB BASIC CONCEPTS

A. Tokenizer and POS Tagging

As a first step in pipeline method of natural language translation, question from the user is tokenized with the help of Stanford parser[14]. It separates the words from query as unique tokens. Part of Speech (POS) tagger take those tokens as input to further process them by extracting the parts of the speech from the words. Parts of speech act as huge support for building up the structured query. Those part of speech include conjunctions, proper nouns or nouns in the sentence [22]. It helps to identify the keywords of the sentence. For example, in the sentence "Number of Provinces in Pakistan", knowing that "Pakistan" is a proper noun and Province is a noun help in recognizing them as keywords of the sentence.

B. Dependency Parsing

Next step in the pipeline is relation extraction of a sentence. It is done by dependency parsing of the sentence. Tokens extracted from tokenized stage are feed as input to the dependency parser. Dependency parser extracts the relationships

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

between keywords identified previously. Such as, "Get me highest rating 3 movies" example has the vagueness of number 3 related to rating or movies. In this step dependency parsing comes to action and find the relation of number 3 with the word movies which is keyword in the sentence. Basic structure of the query is built up on the base of these relationships [12]. Following is the figure. 3 illustrating the dependency parsing.

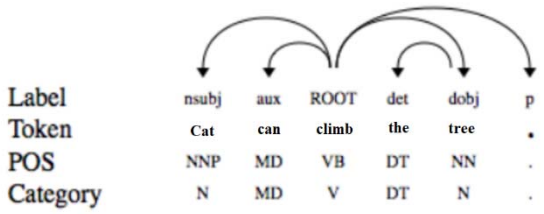

Figure 3. Dependency parsing illustration.

Table 2. Overview of NLIDB with pipeline method concepts.

Concepts	Description
Tokenizer and POS Tagging	Create tokens from sentence and identifying the parts of speech.
Dependency Parsing	Finding the words relationship intra sentence.
Syntactic Parsing	Finding the phrases relationships in a sentence.
Building SQL Query	Consists of entity extraction, Relation operator, Unit conversion and intermediate query.
Ontology building	Building a lexicon of database entities and components.

C. Syntactic Parsing

Besides dependency parser there is another way to find the relationship between words in natural language sentence. It is known with 2 labels such as constituency parser or syntactic parsing. Syntactic parser breaks the sentence into meaningful parts or phrases which highlight the relationship between them. On the other hand dependency parser portrays the links and connections between individual word tokens [23]. Fig 3 in the following shows the illustration of syntactic parsing.

D. Building SQL Query

SQL query building part consists of four subtasks generally. They are explained one by one in the following.

1- **Entity extraction**: Extracting entities information from the database is an important part as to translate a question into query column and table names are important part of knowledge. To get that knowledge a set of column and table names is maintained along with their respective set of synonym words [24]. When a user interacts with a natural language question, synonyms of entity names in tables are identified and converted to the actual entity names in according to the targeted database. For example, if we have a question with "age" in it and we have a column of "date", then "age" will be translated according to the "date" column.

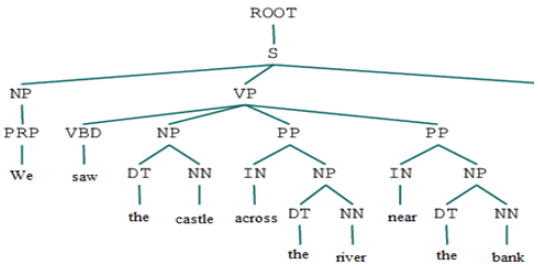

Figure 4. Syntactic parsing illustration.

2- **Relation Operator Extraction**: In this step all the words in the natural language query that express the relation operator are converted into the operators. For example, with a natural language question of "which country has higher import than the china?", higher here represents the comparison operator between the import of different countries. For this purpose, a dictionary of possible operators is build up to handle the issue. Comparison words in the question are directly mapped to the relation operators in the list for the structured queries.

3- **Unit Conversion**: In the database there is usually one standard unit stored for something. User ask for information in their desired units. In order the translate the natural language question into SQL query according to the targeted database, unit conversion is important part. For example, if Natural language question is "Give youngest employ in the department" and database store the date of birth of employees in date

format then that has to be converted into years [25].

4- **Intermediate Query**: In this part of the process complex queries are divided into subqueries. There are complex and difficult queries which otherwise can create vagueness, they are divided into sub tasks to make things more appropriate. For example, if there is a natural language question of "Youngest employee of the month" is divided into "employees of the month" and "youngest employee". This part is mainly done by dependency parser. These sub queries represent the intermediate query in this process.

E. Ontology Building

Building ontology meaning building a dictionary of database components and entities to use as component of external knowledge. There is various way to define the criteria for an ontology, defining hand written rules is one of them. Three basic rules used for this purpose are classes learning, hierarchy learning and properties learning rules. After the normalization process of database, columns and entities are all divided and spread into normalized relations and pieces. Therefore, it is difficult to map the right tokens to the accurate database columns [26]. Building an ontology makes things organized and manageable thus easier to map the semantics used in the natural language question and in the database as shown in figure 4. Following are the five rules category that are mostly utilized for ontology building purpose.

1. Classes Learning rules.
2. Properties Learning rules.
3. Hierarchy Learning rules.
4. Cardinality Learning rules.
5. Instances Learning rule.

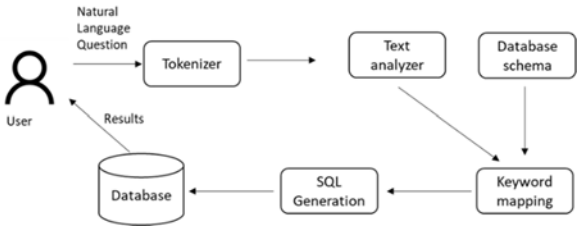

Figure 4. Pipeline method illustration.

2.3. DEEP LEARNING NLIDB SYSTEM BASIC CONCEPTS

A. Encoder

Encoder is the collection of recurrent units such as GRU and LSTM networks working together taking one element at a time from input token sequence to calculate hidden state for that particular component

Concepts	Description
Encoder	Encodes the input sequence in hidden states.
Decoder	Predicts the output based on encoded hidden states.
Word Embedding	Representation of words with context.
Attention Mechanisms	Relationship and dependencies between given components.

based on related information gathered [27]. Calculated hidden state then passed forward for further processing. In text to SQL translation task, tokens of sentence are passed as input to the encoder and hidden state is calculated as output [28].

Table. 3 Overview of NLIDB with deep learning basic concepts.

Encoder vector is the output produced from this unit in the form of hidden state. Hidden state of the input sequence contains the information regarding each element separately in the form of capsule like component as shown in figure 5. All that relevant information capsuled in hidden state helps the decoder to make the right prediction. That hidden state is fed as input to the decoder of the particular model.

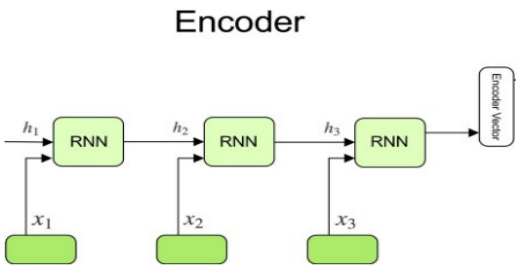

Figure 5. Encoder illustration.

B. Decoder:

Collection of recurrent units that are combined together for the prediction of output. Every recurrent unit takes the hidden states from the encoder as input and produce the output based on that as prediction [29]. Every unit produce hidden state as intermediate form and last unit produce the

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

final output of the model. In cased of text to SQL task, output of decoder is usually a sequence of structured query components [30]. Weights are calculated for all the hidden state at some particular time step. Then SoftMax (a probability distributer) is used to calculate the probability vector to find the final output sentence.

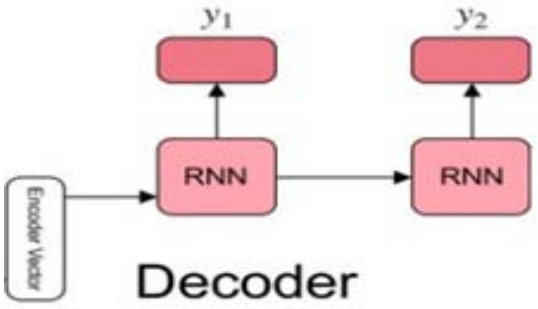

Figure 6. Decoder illustration.

C. Word Embedding:

For text to SQL task with deep learning word embedding is an important part of the process. Achieving the representation of a word with its complete context according to the whole document is main purpose of this procedure. Most used representation of complete document vocabulary is word embedding because of it vide spread capabilities. It has the ability to extract complete context of the words from document, it can capture the relationship between words as well as identify the similarity of semantics and syntax of document [31]. To find the exact meaning of word embedding, it can be said that it is vector for each word particularly. Most important attribute of word embedding is that they generate common representation for the words which synonym meanings for them. It is popular for text to SQL translation tasks particularly with deep learning, because of its impressive performance as distributed representation of text. Glove word embedding is one of the popular off the shelf word embedding library [32].

D. Attention Mechanisms:

Attention mechanisms in deep learning are based on the concept of literal meaning of "attention". Meaning of this word is to focus on something and taking into account especially. Therefore, attention mechanism of deep learning also takes some factors into account especially after calculating them form given data. It is a component of broader system and mainly handles the interdependencies of the given elements [33]. It highlights the relationship of particular components given at a particular time step. Following are common types of attention that are being used generally.

1. Attention mechanism for input-output elements. This is known as general attention mechanism.
2. Self-attention mechanism is the attention between input elements.

For example, for text to SQL task where natural language question is "give me capital of Pakistan" and SQL query for the question is "Select city from City where Country=" Pakistan" and Capital is Yes. For this example, the attention mechanism will calculate most relevant part for SQL query from the input question to enhance the context and accuracy of the predicted query.

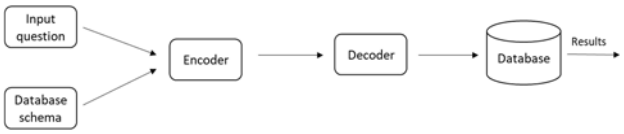

Figure 7. Deep learning NLIDB system illustration.

3. RELATED WORK

As much as review study is need of the area right now, review study on this topic is not being focused recently. Not much work is available in this regard [34]. Available review papers provide basic concept of NLIDB by explaining primary components of NLIDB, beginner systems in this area and advantages/disadvantages of NLIDB in general [35, 36]. More specific and up-to-date review study has been carried on by [37]. Scope of their study is broader and covers general NLIDB systems [38]. Recent literature review study focused on the comparison of NL Interface for SQL and noSql based frameworks. They concluded that 70% of work in NLIDB has been carried out for SQL [39]. Another work effort by [40] focused mainly on only pipeline methods findings and challenges. In our paper we are going to focus on the comparison of NL interface for SQL frameworks with pipeline methods and machine learning methods. In this section we provide a brief literature overview for each of these [41].

Pipeline NLIDBs are constructed by the approach to combine techniques from Natural

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

Language Processing and database areas. Basic of these approach stand on the 4 steps of the process [42]. POS (Part of speech) tagging, text analysis, keyword mapping and generating SQL query finally. Research in this area goes back to sixties when domain specific NLIDB systems were created from manual built up grammar [43, 44]. Recent research in pipeline methods is more general and based on advancements in NLP [1, 45]. Recent works utilize the intermediate representations. Ref [4] Utilized parsed tree in additional processes like Parse Tree node mapper and Parse tree structure adjuster to get more accurate and fine form of Query tree [46]. They also inserted user interaction via Interactive communicator. An ontology based approach was proposed by [47]. They proposed to create a domain specific ontology from given database schema. Map NLQ to intermediate ontology query language and then subsequently translate to the SQL with user interaction to select final query among top ranked candidate queries. Another important contribution is made by [8]. They proposed an iterative synthesis program for parse tree repairing [48]. Proposed system takes a parse tree sketch as input and repair it with iterative synthesize program and finally provides candidate SQL queries to the user. [49] Proposed a method to bridge semantic gap by utilizing SQL query logs for keyword mapping process. Query fragment graph generated from query log improve the understanding of user intention and helps to resolve NL ambiguity [50]. They achieved up to 85% of accuracy with yelp dataset and also covered joins. Work by [25] mainly worked on unit conversion by NLP techniques like POS (part of speech). They handled yes/no questions and "wh" questions only. Basic ground of this work is Dependency grammar semantic parsing like much other work in the area.

As alternative approach to the pipelining methods, deep learning NLIDB systems are being studied. Advances in deep learning inspired and end to end deep learning framework to handle NL queries. One of the pioneer work by [51] to show that Deep Neural Nets can be used to perform "End to End" Translation through Seq2Seq Learning. They demonstrated that LSTM can be used with minimum assumptions, proposing a 2 LSTM (an "Encoder"- "Decoder") architecture to do Language Translation, showing the promise of Neural Machine Translation (NMT) over Statistical Machine Translation (SMT) with a limited vocabulary. [9] Extended Seq2Seq into Seq2SQL model. This model rewards from in-the-loop query execution on a database using a mixed objective, combining cross entropy losses and policy-based reinforcement learning RL. It resolves the issue of queries unordered nature. [7] Proposed SQLNet for the sequence-to-set generation to resolve "order matters" problem in sequence-to-sequence models. It further proposes a novel attention structure called column attention which handles the ambiguity in where clause. Their test results show that it outperforms Seq2SQL by 9 to 13 points on the WikiSQL dataset. Recent work for this area by [52] suggested an integration of dual RNN model for user interaction with SQLNet model as black box for query generation. Proposed system, DialSQL can detected potential errors in generated query and validate by user dialogues with system. Additionally, a simulator also proposed to bootstrap training data for user system dialogue sets.

We present this paper where we can put together and analyze all this work in terms of understanding research challenges and their proposed solutions as well as outcomes and limitations of those solutions. This will provide an overall picture of current areas being focused or areas needs to be focused for text to Sql research.

Table 4 Review papers for NLIDB system

Reference	Study Focus	Limitations
E U and P C 2017	Covered state of the art NLIDB starting from 70s.	Not up to dated, Lack of analysis, Lack of latest work.
Y. Li and D. Rafiei, 2017	Findings and challenges explained.	Only pipeline methods included.
S Dar, I Lali et al. 2019	Analysis of latest work on text to structured queries.	Less focus on text to SQL task specifically
Affolter, Stockinger et al. 2019	Latest NLIDB work analysis	No direct comparison of Deep learning work with pipeline NLIDB systems.

Pipeline NLIDBs are constructed by the approach to combine techniques from Natural Language Processing and database areas. Basic of these approach stand on the 4 steps of the process [42]. POS (Part of speech) tagging, text analysis, keyword mapping and generating SQL query

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

finally. Research in this area goes back to sixties when domain specific NLIDB systems were created from manual built up grammar [43, 44]. Recent research in pipeline methods is more general and based on advancements in NLP [1, 45]. Recent works utilize the intermediate representations. Ref [4] Utilized parsed tree in additional processes like Parse Tree node mapper and Parse tree structure adjuster to get more accurate and fine form of Query tree [46]. They also inserted user interaction via Interactive communicator. An ontology based approach was proposed by [47]. They proposed to create a domain specific ontology from given database schema. Map NLQ to intermediate ontology query language and then subsequently translate to the SQL with user interaction to select final query among top ranked candidate queries. Another important contribution is made by [8]. They proposed an iterative synthesis program for parse tree repairing [48]. Proposed system takes a parse tree sketch as input and repair it with iterative synthesize program and finally provides candidate SQL queries to the user. [49] Proposed a method to bridge semantic gap by utilizing SQL query logs for keyword mapping process. Query fragment graph generated from query log improve the understanding of user intention and helps to resolve NL ambiguity [50]. They achieved up to 85% of accuracy with yelp dataset and also covered joins. Work by [25] mainly worked on unit conversion by NLP techniques like POS (part of speech). They handled yes/no questions and "wh" questions only. Basic ground of this work is Dependency grammar semantic parsing like much other work in the area.

As alternative approach to the pipelining methods, deep learning NLIDB systems are being studied. Advances in deep learning inspired and end to end deep learning framework to handle NL queries. One of the pioneer work by [51] to show that Deep Neural Nets can be used to perform "End to End" Translation through Seq2Seq Learning. They demonstrated that LSTM can be used with minimum assumptions, proposing a 2 LSTM (an "Encoder"- "Decoder") architecture to do Language Translation, showing the promise of Neural Machine Translation (NMT) over Statistical Machine Translation (SMT) with a limited vocabulary. [9] Extended Seq2Seq into Seq2SQL model. This model rewards from in-the-loop query execution on a database using a mixed objective, combining cross entropy losses and policy-based reinforcement learning RL. It resolves the issue of queries unordered nature. [7] Proposed SQLNet for the sequence-to-set generation to resolve "order matters" problem in sequence-to-sequence models. It further proposes a novel attention structure called column attention which handles the ambiguity in where clause. Their test results show that it outperforms Seq2SQL by 9 to 13 points on the WikiSQL dataset. Recent work for this area by [52] suggested an integration of dual RNN model for user interaction with SQLNet model as black box for query generation. Proposed system, DialSQL can detected potential errors in generated query and validate by user dialogues with system. Additionally, a simulator also proposed to bootstrap training data for user system dialogue sets.

We present this paper where we can put together and analyze all this work in terms of understanding research challenges and their proposed solutions as well as outcomes and limitations of those solutions. This will provide an overall picture of current areas being focused or areas needs to be focused for text to Sql research.

4. COMPARATIVE STUDY

4.1. CHALLENGES FOR PIPELINING METHODS NLIDB

Pipeline methods are build up by integrating subtasks together to find the final query. Natural language processing techniques and data science techniques are integrated to build affective framework for the text to SQL task. Work on NLIDB with pipeline methods have been done since 70s but it has its complications linked which prove to be challenges for further progress in the area [52]. Variety of work has been invested to resolve those problems but still accuracy is not enough for industrial usage of the system. Following are the major challenges and issues discussed in detail. Techniques proposed to cope with those issues are also highlighted to explain and bring their shortcomings into light as well.

A. Manual Integration of Techniques:

In pipelining methods Each sub problem is explicitly handled and separate techniques and methods [49]. After every phase output of that particular phase is input for the next one. These are the intermediate states of the input data and understanding of Intermediate representations is critical to manipulate the processes individually. This understanding is critically required to integrate the sub processes effectively for building up the whole NLIDB system [53]. Pipelining methods are combination of techniques from NLP and database communities. Therefore, understanding of both

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

areas is necessary to formulate any variations in processes. Also, integrating various steps manually like Parsed tree, query sketch, ontology building, explicitly defining semantic coverage have more error margin, and each step needs separate effort and attention. Besides all problems, this situation is being utilized to improve system performance.

1. **Intermediate states utilization:** To improve system performance with pipeline methods, work focus is Innovations and experimenting with intermediate representations of NL to SQL. Such as Splitting key word mapper, integrating novel scoring methods, ontology based Intermediate query language, Parse tree repairing [4, 8, 47, 49]. To Utilizing these intermediate states more steps, need to be added or split existing once into multiple. Complication of process increase with every additional step which increase compatible considerations and cost as well.
2. **User Interaction for individual steps:** User interaction for NLIDBs help to understand user's intention more clearly. In pipeline method user interaction can be integrated at any stage. Any intermediate representation can be validated from user [4, 47, 49]. For this approach, expert users are required who can understand the intermediate state. Although in mentioned systems they proposed description along with intermediate states but that cannot be generally understandable for layman users accessing the database.

4.2. SYSTEM EXPANSION LIMITATIONS

Pipeline Methods NLIDB have limited scope of system expansion like cross domain or cross language system. As they work based on carefully designed rules and domain based ontologies. Rule based systems are especially difficult for cross language expansions[52].

1. **Working directly with DB schema:** Working directly with database schema means incorporating the database schema in the process directly. This way whole schema information including relations and entities become part of the process and unseen schema can also get handled. Recent efforts have been made to make pipeline NLIDBs domain independent by involving database schema directly in the process. It makes system more portable [4]. But it does not capture the full domain semantics hence not fully capable for cross domain application**.**
2. **Ontology Driven Systems:** Ontology based NLIDB shown to be useful for cross domain application. [47] displayed up to 88.9% recall on FIN dataset with an ontology driven NLIDB . This work is implementable if ontology to database mapping is provided. Generating ontology from database itself is an open research problem

4.3. CHALLENGES FOR DEEP LEARNING NLIDB

A. Dataset Unavailability:

Recent state of the art Deep learning NLIDB [7, 54] show promising results in the text to SQL conversion with large volumes of training data set. But as pointed out by [49] with small set of training data deep learning methods are impractical [55]. Until this point we have few labelled datasets with basic level of simple queries [56]. Hence state of the art systems has been trained and tested on simple queries datasets only i.e with no joins [57, 58]. Following is the discussion of contributes that have been made to solve these issues with the trade-offs of cost, time or practicality of method.

- **a- Manually synthesizing NL SQL pairs:** Manually synthesizing means creating the SQL from the Natural language questions manually for training dataset as well as test dataset. This is a long and hectic process to create a dataset from scratch that too manually. Besides its difficulty some recent efforts have been made to create manual labelled data of NLQ-SQL [6, 59]but it is costly and time consuming process.
- **b- NL Generation from SQL:** SQL query logs can provide the generally asked queries of a particular database. Reverse generating the natural language question from those queries can provide with labeled dataset. Driving NL questions from user SQL queries is another way to create labelled data [60][18] but data is not real time which results in biased experimental results [56, 61]. Hence this approach is neither efficient nor practical [62]. This is another hectic method and also generated dataset will be biased. Natural language questions asked by laymen will be different from the questions generated by the experts from the queries. Therefore, this generated cannot represent the complexity level of real time data.
- **c- Use of Transfer learning with limited domain data:** Training the model on the dataset of a different but available domain and then utilizing pre-trained model for the targeted domain which has small dataset is

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

Research Challenges	*Approaches in Practice*	*Limitations*
Deep Learning NLIDB		
Labelled Dataset Unavailability	Manual Labelling	Cost and time issues
	NL generation from SQL	Biased Dataset
	Transfer learning with another domain dataset	Cost and time doubles
	Data augmentation	Limited expansion of data
	Simulators usage	data Not realistic
Black Box Property of Deep Learning	Input Representation Pre-processing	Time and cost without real knowledge of affect ratio
	Output validation from user	Dialogue data unavailability
		Vaguely defined criteria of Human expertise
Pipelining NLIDB		
Manual Integration of Techniques	Intermediate states utilization	Increased complexity of Process, More considerations for process compatibility
	User Interaction for individual steps	Experts required
System Expansion Limitations	Working directly with DB schema	Full domain semantics not captured
	Ontology Driven Systems	Generating ontology to DB mapping

the intuition behind this method. [63, 64] Displayed an approach to develop deep learning NLIDB for domains with small datasets. This improved the results but still not enough to be used practically [65]. Also cost and time wise it is not practical to train a model on one domain before getting in use for target domain [66]. This way it will increase training time for training the model on 2 datasets as well as parsing errors can increase because of semantic issues not covered properly for the targeted domain.

d- Augmentation techniques: Augmentation means expanding the dataset from given dataset by duplicating examples with small variations. However, this method generates biased datasets like manual synthesizing. Recent end-to-end deep learning systems [7, 9, 59] show the great promise of learning from large volumes of NLQ-SQL pairs. However, manually creating labelled NLQSQL pairs is costly and time-consuming. Despite recent efforts to synthesize NLQ-SQL pairs [6, 59] or derive them from user descriptions of SQL queries[60], obtaining realistic labelled data remains an open research challenge. As a result, state-of the-art deep learning systems [7, 9] have thus far only been tested on datasets of simple NLQs requiring no join.

B. Deep Learning Black Box Property:

Having that deep learning systems implicitly tackle many challenges, at the same time the lack of understanding of the mechanisms behind Neural Network's effectiveness limits further improvements on the architecture [67]. Therefore, input representation manipulation [19] or output post processing [15] are the possible ways to get improved results for NLIDB systems. Network selection from available deep learning networks is another factor for improving results as experimented by [16], but it has limited options to offer.

a- Input Representation Pre-processing: Input representation pre-processing displayed by [63] includes Gan-based augmentation method to expand data set which contributed in improving results despite the fact that it is not much realistic [68, 69]. Data augmentation or recombination [70], keyword scoring vector integration are popular ways to change input representation for better results in deep learning based NLIDBs [71]. Despite the fact that these approaches can change the output ratios, it's still difficult to understand which factor affected the part of process most and in what ways [72].

b- Output validation from user: As understanding user intention and influencing the system with this factor is critical for NLIDB [73]. For this this purpose incorporating Human in the loop structure in

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

NLIDB as pre/post process is another popular approach for improving results [4, 8, 52]. Human in the loop with deep learning invites 2 more challenges; how to get dialogue data for training and evaluation of the system [74]. [52] solved these issues by using a simulator but again that cannot reflect real time data. Also performance of the systems with human in loop rely mostly on involved human's expertise, which cannot be defined concretely [75].

5. CONCLUSION

This comparative study is based on the literature review available for NLIDB with pipeline methods and deep NLISB systems with deep learning. In this paper we have briefly explained the existing work for both approaches as well as their background knowledge of basic ideas and concepts have been explained in detail. All these perspectives are provided to portray more clear picture of current work situation in this area. Observing the current trends and work ideas a comparative analysis for both approaches have been provided. With all the available literature and background study we have concluded that NLIDB with pipeline methods are still in trend somewhat but not popular as much as deep learning NLIDB systems. Reason is there manual integration of each step that increases the error chances overall. On the other hand, Deep learning NLIDBs are proving to be need of time because of their quick learning and great potential of further progress and expansion. Deep learning NLIDB have more potential of expansion across domains and different languages as well as it is automating the whole process which makes it less complicated to study and experiment. Besides all this benefits deep learning NLIDB face a major issue of data unavailability which is a huge hurdle for further progress in the field currently. Therefore, transfer learning and meta learning are becoming trend of the research in this area because of their handling with small datasets.

6. ACKNOWLEDGEMENT

This research was supported by the MISP(Ministry of Science, ICT), Korea, under the National Program for Excellence in SW)(2018-0-00192) supervised by the IITP(Institute of Information & communications Technology Planning & Evaluation)"(2018-0-00192).

REFERENCES

[1]. C. Finegan-Dollak et al., "Improving text-to-SQL evaluation methodology," in ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers), 2018, vol. 1, pp. 351-360, doi: 10.18653/v1/p18-1033. [Online]. Available:

[2]. Androutsopoulos, G. D. Ritchie, and P. Thanisch, "Natural language interfaces to databases-an introduction," arXiv preprint cmp-lg/9503016, 1995.

[3]. B. Bogin, M. Gardner, and J. Berant, "Representing schema structure with graph neural networks for text-to-sql parsing," arXiv preprint arXiv:1905.06241, 2019.

[4]. F. Li and H. Jagadish, "Constructing an interactive natural language interface for relational databases," Proceedings of the VLDB Endowment, vol. 8, no. 1, pp. 73-84, 2014.

[5]. P. He, Y. Mao, K. Chakrabarti, and W. Chen, "X-SQL: reinforce schema representation with context," arXiv preprint arXiv:1908.08113, 2019.

[6]. S. Iyer, I. Konstas, A. Cheung, J. Krishnamurthy, and L. Zettlemoyer, "Learning a neural semantic parser from user feedback," arXiv preprint arXiv:1704.08760, 2017.

[7]. X. Xu, C. Liu, and D. Song, "SQLNet: Generating structured queries from natural language without reinforcement learning," 2018.

[8]. 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, pp. 1-26, 2017.

[9]. V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," arXiv preprint arXiv:1709.00103, 2017.

[10]. B. Bogin, M. Gardner, and J. Berant, "Global reasoning over database structures for text-to-sql parsing," arXiv preprint arXiv:1908.11214, 2019.

[11]. D. Choi, M. C. Shin, E. Kim, and D. R. Shin, "RYANSQL: Recursively Applying Sketch-based Slot Fillings for Complex Text-to-

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

SQL in Cross-Domain Databases," arXiv preprint arXiv:2004.03125, 2020.

[12]. V. Wudaru, N. Koditala, A. Reddy, and R. Mamidi, "Question Answering on Structured Data using NLIDB Approach," in 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), 2019: IEEE, pp. 1-4.

[13]. F. Li and H. V. Jagadish, "NaLIR: an interactive natural language interface for querying relational databases," in Proceedings of the 2014 ACM SIGMOD international conference on Management of data, 2014, pp. 709-712.

[14]. A.-M. Popescu, A. Armanasu, O. Etzioni, D. Ko, and A. Yates, "Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability," in COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics, 2004, pp. 141-147.

[15]. Y. Li, H. Yang, and H. Jagadish, "NaLIX: A generic natural language search environment for XML data," ACM Transactions on database systems (TODS), vol. 32, no. 4, pp. 30-es, 2007.

[16]. Y. Li, H. Yang, and H. Jagadish, "Constructing a generic natural language interface for an XML database," in International Conference on Extending Database Technology, 2006: Springer, pp. 737-754.

[17]. N. Stratica, L. Kosseim, and B. C. Desai, "Using semantic templates for a natural language interface to the CINDI virtual library," Data & Knowledge Engineering, vol. 55, no. 1, pp. 4-19, 2005.

[18]. E. M. Eisman, M. Navarro, and J. L. Castro, "A multi-agent conversational system with heterogeneous data sources access," Expert Systems with Applications, vol. 53, pp. 172-191, 2016.

[19]. G. Cai, H. Wang, A. M. MacEachren, and S. Fuhrmann, "Natural conversational interfaces to geospatial databases," Transactions in GIS, vol. 9, no. 2, pp. 199-221, 2005.

[20]. N. Yaghmazadeh, Y. Wang, I. Dillig, and T. Dillig, "Type-and content-driven synthesis of SQL queries from natural language," arXiv preprint arXiv:1702.01168, 2017.

[21]. C. Wang, M. Brockschmidt, and R. Singh, "Pointing out SQL queries from text," 2018.

[22]. D. H. Warren and F. C. Pereira, "An efficient easily adaptable system for interpreting natural language queries," American journal of computational linguistics, vol. 8, no. 3-4, pp. 110-122, 1982.

[23]. H. Ghassani and T. E. Widagdo, "Access to Relational Databases Using Interrogative Sentences in Indonesian Language," in 2018 5th International Conference on Data and Software Engineering (ICoDSE), 2018: IEEE, pp. 1-6.

[24]. C. Wang, A. Cheung, and R. Bodik, "Synthesizing highly expressive SQL queries from input-output examples," in Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2017, pp. 452-466.

[25]. F. Reinaldha and T. E. Widagdo, "Natural language interfaces to database (NLIDB): Question handling and unit conversion," in 2014 International Conference on Data and Software Engineering (ICODSE), 2014: IEEE, pp. 1-6.

[26]. K. Affolter, K. Stockinger, and A. Bernstein, "A comparative survey of recent natural language interfaces for databases," The VLDB Journal, vol. 28, no. 5, pp. 793-819, 2019.

[27]. W. Wang, Y. Tian, H. Xiong, H. Wang, and W.-S. Ku, "A transfer-learnable natural language interface for databases," arXiv preprint arXiv:1809.02649, 2018.

[28]. J. Zeng et al., "Photon: A Robust Cross-Domain Text-to-SQL System," arXiv preprint arXiv:2007.15280, 2020.

[29]. Z. Yao, Y. Su, H. Sun, and W. T. Yih, "Model-based interactive semantic parsing: A unified framework and a text-to-SQL case study," in EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference, 2020, pp. 5447-5458.

[30]. S. Yavuz, I. Gur, Y. Su, and X. Yan, "What it takes to achieve 100% condition accuracy on wikisql," in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 1702-1711.

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

[31]. C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations, 2014, pp. 55-60.

[32]. J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532-1543.

[33]. Y. Sun et al., "Semantic parsing with syntax- and table-aware sql generation," arXiv preprint arXiv:1804.08338, 2018.

[34]. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[35]. J. Patel and J. Dave, "A Survey: Natural Language Interface to Databases," International Journal of Advance Engineering and Research Development (IJAERD), 2015.

[36]. E. Reshma and P. Remya, "A review of different approaches in natural language interfaces to databases," in 2017 International Conference on Intelligent Sustainable Systems (ICISS), 2017: IEEE, pp. 801-804.

[37]. Y. Li and D. Rafiei, "Natural Language Data Management and Interfaces," Synthesis Lectures on Data Management, vol. 10, no. 2, pp. 1-156, 2018.

[38]. L. Dong and M. Lapata, "Coarse-to-fine decoding for neural semantic parsing," arXiv preprint arXiv:1805.04793, 2018.

[39]. H. S. Dar, M. I. Lali, M. U. Din, K. M. Malik, and S. A. C. Bukhari, "Frameworks for Querying Databases Using Natural Language: A Literature Review," arXiv preprint arXiv:1909.01822, 2019.

[40]. Y. Li and D. Rafiei, "Natural language data management and interfaces: Recent development and open challenges," in Proceedings of the 2017 ACM International Conference on Management of Data, 2017, pp. 1765-1770.

[41]. Z. Dong, S. Sun, H. Liu, J.-G. Lou, and D. Zhang, "Data-Anonymous encoding for Text-to-SQL generation," in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 5408-5417.

[42]. A. Elgohary, S. Hosseini, and A. H. Awadallah, "Speak to your Parser: Interactive Text-to-SQL with Natural Language Feedback," arXiv preprint arXiv:2005.02539, 2020.

[43]. E. D. Sacerdoti, "Language access to distributed data with error recovery," SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1977.

[44]. W. Woods, "The lunar sciences natural language information system," BBN report, 1972.

[45]. A. Giordani and A. Moschitti, "Generating SQL queries using natural language syntactic dependencies and metadata," in International Conference on Application of Natural Language to Information Systems, 2012: Springer, pp. 164-170.

[46]. J. Guo et al., "Towards complex text-to-sql in cross-domain database with intermediate representation," arXiv preprint arXiv:1905.08205, 2019.

[47]. 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, 2016.

[48]. T. Shi, K. Tatwawadi, K. Chakrabarti, Y. Mao, O. Polozov, and W. Chen, "Incsql: Training incremental text-to-sql parsers with non-deterministic oracles," arXiv preprint arXiv:1809.05054, 2018.

[49]. C. Baik, H. V. Jagadish, and Y. Li, "Bridging the semantic gap with SQL query logs in natural language interfaces to databases," in 2019 IEEE 35th International Conference on Data Engineering (ICDE), 2019: IEEE, pp. 374-385.

[50]. R. Zhang et al., "Editing-Based SQL Query Generation for Cross-Domain Context-

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

Dependent Questions," arXiv preprint arXiv:1909.00786, 2019.

[51]. I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in Advances in neural information processing systems, 2014, pp. 3104-3112.

[52]. Gur, S. Yavuz, Y. Su, and X. Yan, "DialSQL: Dialogue based structured query generation," in ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 2018, vol. 1, pp. 1339-1349.

[53]. J. Sen, A. R. Mittal, D. Saha, and K. Sankaranarayanan, "Functional Partitioning of Ontologies for Natural Language Query Completion in Question Answering Systems," in IJCAI, 2018, pp. 4331-4337.

[54]. P.-S. Huang, C. Wang, R. Singh, W.-t. Yih, and X. He, "Natural language to structured query generation via meta-learning," arXiv preprint arXiv:1803.02400, 2018.

[55]. T. Yu et al., "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018, 2020, pp. 3911-3921.

[56]. Z. Lu, H. Li, and B. Kao, "Neural enquirer: learning to query tables in natural language," IEEE Data Eng. Bull., vol. 39, no. 3, pp. 63-73, 2016.

[57]. H. Liu, L. Fang, Q. Liu, B. Chen, J. G. Lou, and Z. Li, "Leveraging adjective-noun phrasing knowledge for comparison relation prediction in text-to-SQL," in EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference, 2020, pp. 3515-3520.

[58]. B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, "Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers," arXiv preprint arXiv:1911.04942, 2019.

[59]. P. Utama et al., "An end-to-end neural natural language interface for databases," arXiv preprint arXiv:1804.00401, 2018.

[60]. F. Brad, R. Iacob, I. Hosu, and T. Rebedea, "Dataset for a neural natural language interface for databases (NNLIDB)," arXiv preprint arXiv:1707.03172, 2017.

[61]. W. Wang, "A cross-domain natural language interface to databases using adversarial text method," in CEUR Workshop Proceedings, 2019, vol. 2399.

[62]. U. Brunner and K. Stockinger, "ValueNet: A Neural Text-to-SQL Architecture Incorporating Values," arXiv preprint arXiv:2006.00888, 2020.

[63]. H. Xiong and R. Sun, "Transferable natural language interface to structured queries aided by adversarial generation," in 2019 IEEE 13th International Conference on Semantic Computing (ICSC), 2019: IEEE, pp. 255-262.

[64]. C. Lawrence and S. Riezler, "Nlmaps: A natural language interface to query openstreetmap," in Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, 2016, pp. 6-10.

[65]. D. Lee, J. Yoon, J. Song, S. Lee, and S. Yoon, "One-shot learning for text-to-sql generation," arXiv preprint arXiv:1905.11499, 2019.

[66]. K. Lin, B. Bogin, M. Neumann, J. Berant, and M. Gardner, "Grammar-based neural text-to-sql generation," arXiv preprint arXiv:1905.13326, 2019.

[67]. Y. Ming et al., "Understanding hidden memories of recurrent neural networks," in 2017 IEEE Conference on Visual Analytics Science and Technology (VAST), 2017: IEEE, pp. 13-24.

[68]. T. Guo and H. Gao, "Content Enhanced BERT-based Text-to-SQL Generation," arXiv preprint arXiv:1910.07179, 2019.

[69]. T. Yu, Z. Li, Z. Zhang, R. Zhang, and D. Radev, "Typesql: Knowledge-based type-aware neural text-to-sql generation," arXiv preprint arXiv:1804.09769, 2018.

[70]. R. Jia and P. Liang, "Data recombination for neural semantic parsing," arXiv preprint arXiv:1606.03622, 2016.

[71]. G. Huilin, G. Tong, W. Fan, and M. Chao, "Bidirectional Attention for SQL Generation," in 2019 IEEE 4th International Conference on Cloud Computing and Big

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

Data Analysis (ICCCBDA), 2019: IEEE, pp. 676-682.

[72]. W. Hwang, J. Yim, S. Park, and M. Seo, "A comprehensive exploration on wikisql with table-aware word contextualization," arXiv preprint arXiv:1902.01069, 2019.

[73]. I.A. Hosu, R. C. A. Iacob, F. Brad, S. Ruseti, and T. Rebedea, "Natural language interface for databases using a Dual-Encoder model," in Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 514-524.

[74]. A. Kelkar, R. Relan, V. Bhardwaj, S. Vaichal, and P. Relan, "Bertrand-DR: Improving Text-to-SQL using a Discriminative Re-ranker," arXiv preprint arXiv:2002.00557, 2020.

[75]. D. Lee, "Clause-Wise and Recursive Decoding for Complex and Cross-Domain Text-to-SQL Generation," arXiv preprint arXiv:1904.08835, 2019.