International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-4, April 2019

Optimizing Natural Language Interface for Relational Database

Darshil Shah, D Vanusha

Abstract: ***This paper is research on developing a natural language interface for the relational databases, (RDBMS), which takes in the natural language in the form of english like statements as inputs from the operator or user and generates answers in the form of SQL(Structured Query Language) queries. Firstly, the input in the form of natural language is parsed using a semantic grammar and later, it is translated into a SQL query by the use of NLP (Natural Language Processing). The interface makes use of RNN (Recurrent Neural Networks) to convert the natural language input given by the user to the Structured Query language. The network is trained on large number of datasets from the WikiSQL, which makes sure that the accuracy of the model is good enough for real time use in the industry and also for the personal use. Stanford NLP and self-made datasets are also used to an extent to enhance the efficiency and optimise the query formation process. Finally, a database management system is used to find the result set with its own specialized optimization and planning techniques. The database is implemented in the JSON format to reduce the query processing time when the database is too large. The database is also organised in a tiered architecture format so that the query execution time is reduced.***

Index Terms: ***Natural Language Processing, Recurrent Neural Networks, Reinforcement Learning, Relational Data, Structured Query Language, WikiSQL***

I. INTRODUCTION

The Global IT insurgency in the last few years has caused in a grand-scale digitization and generation of data, making it available to millions of users in the form of DBMS, also the data is considered the new currency. However, interacting and using these databases demands and requires a strong comprehensiveness of query languages such as Structured Query Language (SQL), or any other language used fr interaction with the database, which, while being powerful and an efficient way for interacting with the databases is also difficult to learn especially for a person coming from a nontechnical person. This is often regrettably beyond the bounds of the programming competence of a large number of end users. Thus, limiting to the effective semantic parser that are capable of translating natural language statements into the logical forms like queries has been something like a longstanding objective. There already have been many parsers that rely on the straightforward rule based query generation and formation from input such as the natural language, but they stop and fail to give the efficient outputs when the queries start becoming complicated and convoluted like those involving nested subqueries and joins that involves the more than one table data accessing, this deep learning neural network method for query generation from the English like natural language is however a little complex and out of the box approach for this, the deep learning model requires a large number of datasets to achieve the appreciable and efficiency, but once the model has been trained and deployed properly it is capable of converting the natural language input from the user to the corresponding SQL queries involving convulsions and complications i.e. queries with multiple tables also, which the rule based SQL generators fail to do so, since they are always being limited to the rule with which they are developed, and hence turn out to be very inefficient when used with something for which there hasn't been a well defined rule.

There are presently two methods to go from the NL to SQL, they are as below semantic parsing and neural networks. Semantic parsing is a method to convert and translate text to a formal meaning portraying in the logical forms or in the form of structured queries. There already have been many works in the field of semantic parsing. Most of these previously known and implemented systems rely on the greater-quality lexicons, realm representation-specific features and may not generalize. This is why in this work, the focus is on neural network approaches to handle the NL to SQL tasks, which require less use of hard-coded rule that drive the task for conversion into SQL rather they rely on cognitive and self improvising techniques such as deep learning. Recently, a large dataset on NL to SQL has been released by Salesforce: termed as WikiSQL. It is a collection of 80,654 hand-written instances of natural language questions, SQL queries and SQL tables extracted from 24,241 HTML tables from Wikipedia. WikiSQL dataset is larger than all the other sources for datasets combined land of-course larger than previous semantic parsing datasets, which makes it interesting for data-hungry neural networks. The major requirement for constructing an efficient and accurate neural network is to provide a large amount of data to the model to train on. Wiki SQL provides enough datasets to get the considerable accuracy for the model. Along with the deep learning and cognitive techniques the project also uses NLP to a large extent in order to process the user's input. NLP is a theory-motivated range of computational methods for the automated study and depiction of human natural language.

Manuscript published on 30 April 2019.
* Correspondence Author (s)
Darshil Shah*, Dept. of Computer Science, SRM IST, Chennai, India
D Vanusha, Dept. of Computer Science, SRM IST, Chennai, India

The development and research in the field of NLP dates back to the era when punch cards and batch processing were used, in which the parsing and analysis of sentence take as long as tens of minutes, to the era of Google and supercomputing, in which millions of web pages and requests can be processed in the order of milliseconds. NLP allows the systems to process a great range of natural language related tasks across all the hierarchical levels, right from Part-Of-Speech (POS) tagging and parsing, to machine translation and dialogue processing. Recursive Neural Network (RNN) that has turn out to be accurate and efficient when it comes to constructing, representing and processing sentence. However, the RecursiveNN captures the semantics of a sentence using a semantic tree architecture. Its performance largely relies on the performance of the semantic tree structure. Moreover, developing such a textual tree has a time complexity of at least O(n^2), where n represents the length of the text. This would be too inefficient when the model encounters a long statement or gets a whole document to process. Also, the relationship among multiple statements is very difficult and complex to represent in the semantic tree structure. Hence, Recursive NN is in-efficient and therefore unsuitable for processing and parsing very long statements or large text files. Another model, that has a linear time complexity O(n), is the Recurrent Neural Network (RecurrentNN). This model processes a statement word by word and saves the semantics of all the earlier parsed sentences or text in a hidden layer whose size is predetermined during the modeling of the network. RNN is better at capturing the contextual text information, compared to the Recursive Neural Network with an added advantage of lesser time complexity. This could make the use of Recurrent Neural Network favorable to acquire semantics of long statements and long texts. However, the RNN is a biased model, in which the later word hold more weight to the final result of parsing than earlier words. Due to this type of text processing architecture the RNN sometimes produce less accurate result for the larger sentences and large text files, since the key components could appear anywhere in a document rather than at the end. Our project aims at providing the user with a web interface through which he can interact with the model, submit the natural language query to the model, which then gets converted to SQL and finally the output is shown to the user through the web portal. The complete architecture and details of the implementations has been explained in the further sections of the project in detail. Traditionally, development work in getting data from RDBMS often follows one of two paths: the keyword-based method and the structured query method. Both methods have their own pros and cons. The structured query method, while expressive and powerful, is not simple and easy for normal end users. The keyword-based method is favorable to use, but it has its own limitations when it comes to processing the large sentences. In opposition, NLP has both pros to a large extent: so that even the normal end users are capable in expressing the complicated query in the form of natural language. Therefore providing the supporting for natural language queries is often regarded as the ultimate goal for a database query interface. However, advancing has been slow paced, even as in general the NLP systems have improved over the span of years. This is primarily due to the difficulty of converting the user query structure to the actual schema structure represented in the most of the database systems By resolving this complexity, we have eradicated the biggest difficulty in the natural language querying of database. In the project, we present Natural Language Interface for Relational Database (NaLIR), a generalized and interactive natural language interface for working with the RDBMS. In NaLIR, a plain english language sentence, which can be quite complicated and complex in logic, is taken as natural language input. This query is first converted into an SQL query, which may contain nesting, aggregation, and number of types of joins, along with the other things. Then, a RDBMS is used to execute the hence generated SQL query and return the result set back to the user. For example, the input can be like “return the id of writers in database area, whose books have the most gross sale", without knowing about the structure of the columns and organizational knowledge of records in the database. An efficient NaLIR should do the work of a database programmer (dba): i.e. the interface should be able to internally convert the user’s input in the form of the natural language to the appropriate SQL, execute it against the database and finally return the result set back to user obtained from the database. Then, the database admin tells his understanding, first for some inconclusive words back to the user in order to avert the misinterpretation resulting into wrong query formation. After that the user rectifies the dba’s understanding and if there are some changes to be made then they are made by the user at this stage, the data base administrator will form the SQL query, execute it against the database and finally give the record sets back to the user. In the first step, we make use of a natural language parser to get the parse tree of the input natural language statement. In the next stride, we convert the linguistic understanding represented in the form of parse tree to a database’s understanding by comparing and fixing proximity of the arrangement in the parse tree to proximity of corresponding database. In order to ensure that the sentence is correctly parsed and interpreted by the parser, we explain the database’s understanding back to the user in natural language, and at the end after the user verifies the correct processing of his input sentence, the understanding is then converted in to an SQL query statement and evaluated by a RDBMS.

II. RECURRENT NEURAL NETWORK

This project uses the Recurrent neural network for NLP, hence this section briefly discusses the Recurrent Neural Network, its architecture and overview. A recurrent neural network (RNN) is a type of neural networks in which connections between nodes are arranged in a directed graph along with a temporal order. This enables it to display a temporal changing behavior. Dissimilar to a feed forward neural networks, Recurrent Neural Networks can use their internal hidden states which acts like the memory cells to operate on the sequences of sentence or text as inputs.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

This makes the Recurrent neural network ideal for applications like speech and handwriting recognition. There are two types of neural networks one is finite impulse recurrent neural network and the other is indefinite impulse neural networks. A finite impulse recurrent network one having a directed acyclic graph that can be replaced with feed forward neural network, whereas an infinite impulse recurrent network is a directed cyclic graph that can not be converted in to a feed forward neural network. Both of them have the memory states in which they store the previously parsed part of the text and use it to combine it with the later part of sentence in order to devise the correct and accurate meaning of the complete text. - xt is the input given to the network at time t. xt-1 represents the previous phrase or word of the statement. and let ht represent the hidden state at time t. The output from this state will be non linear and represented using the activation function such as tanh or the rectified linear unit. ht-1 is calculated from the previous hidden layer's state usually it is set to zero, let yt represent the output at the time t.

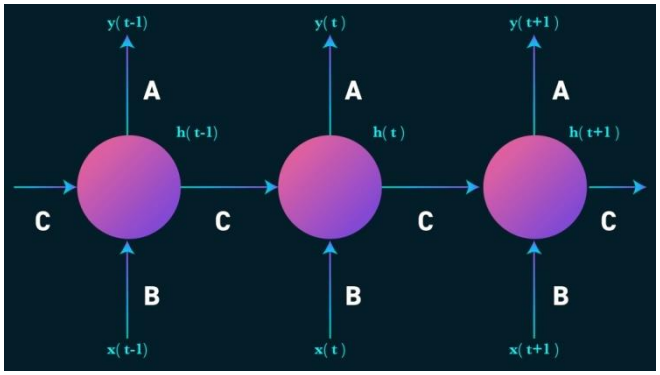

Fig 1. Three layer recurrent neural network which is unrolled to understand the inner iterations

The Recurrent Neural network in the above figure has equal evaluation at every step considering the weights as A, B and C but the inputs differ at each time step making the process quick and simpler. It remembers only the previous and not the words before it acting like a memory. Long Short Term

Memory units take the inputs from the previous hidden layer ht-1 and the current input xt to determine the output. Main

function of these Long Term Short term Cells is to determine what to keep in memory and what to discard from

the memory. All three the value of past state, value of current memory and the value of current input are used in a cohesion to predict the next output.

III. NATURAL LANGUAGE PROCESSING

Natural Language Processing is the capability of a system to comprehend, process and work upon the human like language. NLP can be used to derive meaning from text and analyze it. There is huge degree of information kept in the form of text files, like for example vehicles' records. Before the advancement in deep learning based natural language processing models, this data and information were not accessible to computer-assisted analysis and could not be studied in an organized manner. But now with the advanced and highly accurate models of NLP being developed it is possible to design a model that could easily understand and process large text files having the free text in them. With the advancement in the deep learning based- NL models there have been opening of all new horizon of applications for it such as translation of text/input in other languages, text summarization, semantic comprehension etc. NL is a specialized application of Artificial Intelligence that aims to facilitate systems to comprehend and work with human languages, Although there have been many advancements in the field of Natural language but still the model can't be trusted one hundred per cent and at times the model also fails to understand what the text wants to convey.

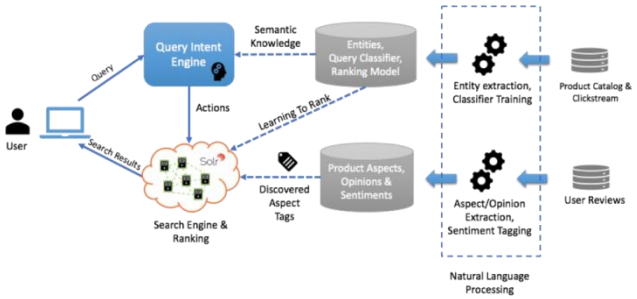

Fig. 2. Basic architecture of NLP using machine learning is shown in the figure

IV. DATASETS

The primary source of datasets for training and testing the natural language interface to relational database are WikiSQL dataset provided by the salesforce, below sub-sections summarizes it in brief.

A. WikiSQL

WikiSQL is a corpus containing questions and its corresponding SQL queries, along with all the SQL tables. Figure 3 shows the sample example that contains a table, a question and corresponding SQL query, WikiSQL is the largest handwritten dataset to date — it is larger than all the other semantic parsing datasets, considering the number of examples and also the number of tables. The queries in the WikiSQL datasets spans through multiple tables and this adds another level of complexity in designing the model: the model should be able to generalize both new queries and also to the new table formats. WikiSQL consists of the rational data taken from web. This is indisputable in the distributions of the number of columns, the set of questions, and the corresponding set of queries.

Table

Player	No.	Nationality	Position	Years in Toronto	School/Club Team
Antonio Lang	21	United States	Guard-Forward	1999-2000	Duke
Voshon Lenard	2	United States	Guard	2002-03	Minnesota
Martin Lewis	32.44	United States	Guard-Forward	1996-97	Butler CC (KS)
Brad Lohaus	33	United States	Guard-Forward	1996	Iowa
Art Long	42	United States	Guard-Forward	2002-03	Cincinnati

Question: Who is the player that wears number 42?

SQL: SELECT player WHERE no.=42

Result: Art Long

Fig. 3. An example of WikiSQL task

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

The above figure shows us an example of table, question and corresponding query from the WiliSQL data-set. In the table "player_info" which consists of column like player, player's number, nationality, position, number of years in toronto and school or club team. The question "Who is the player that wears number 42" is depicted in the example and its corresponding SQL query is "SELECT player WHERE no.=42" and the result is Art Long. Like the above example the WikiSQL consists of large number of such examples where the queries and questions actually span across multiple tables e.g. including joins and nested sub queries in to the query. The tables, their paraphrases, and SQL queries are randomly slotted into train, develop, and test splits, such that each table is present in exactly one split. In addition to the raw tables, queries, results, and natural utterances, we also release a corresponding SQL database and query execution engine.

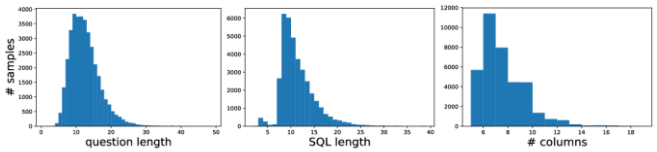

Figure 5: Distribution of numbers of columns, tokens per question, and tokens per query in WikiSQL.

Fig. 4. Columns, tokens and tokens per query distribution in WikiSQL

V. ARCHITECTURE OF THE SYSTEM

The entire project comprises of 3 main modules, front-end, back-end and database. The front end module is responsible for taking in the user's input in the form of natural language and forward it to the backend, that processes it. Backend module accepts the incoming inputs from the front end and processes it to generate the SQL query. The back-end comprises two main subparts. First is responsible for removing the stop words and do the Parts of Speech tagging on the sentence. The next sub module processes this pos tagged output to generate the SQL query. The generated SQL query is then executed against the database and the output is hen presented on the front-end to the user. The figure below shows the entire architecture of the system.

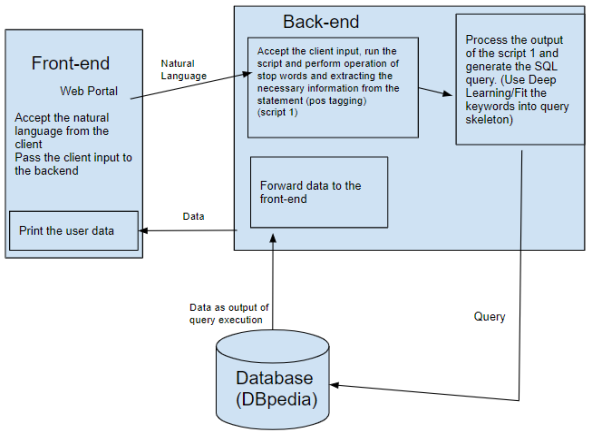

Fig. 5. Architecture diagram of Natural Language interface for relational database

Below sub-sections summarises the individual module architecture.

A. Query Generator Module

The below flow chart shows the internal working and processing of the back-end part. The user's input will be pos tagged after parsing and removing the stop words from the sentence. On the POS tagged data the metadata consisting of the database, columns and tables will be used for pattern recognition. Pattern recognition is essentially a Recurrent Neural network that will be trained over large number of datasets so that it can generate the sql query from the pos tagged data efficiently and accurately.

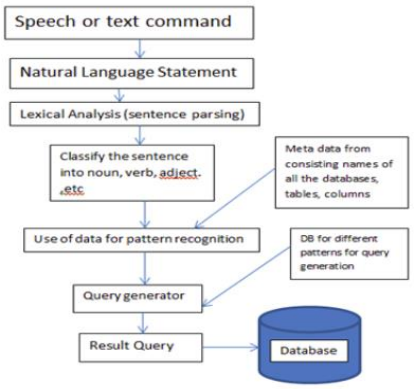

Fig. 6. Query Genrating Module Architecture flow diagram

B. Database Module

The database is organised in the hierarchical structure in the order of database layer, table level and record level. This hierarchical structure allows the easy and fast retrieval of data when data is enormous. In order to further bring down the query execution time primary indexing is also done on the data of the database. The below diagram show the organization of the database.

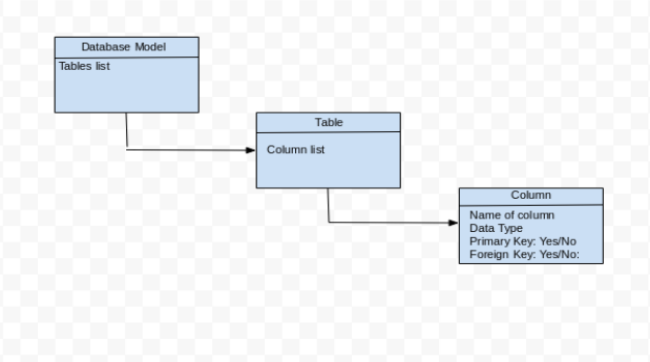

Fig. 7. Database Module Architecture

It's a known fact that the order of the constraints affects the performance of a sequence-to-sequence-styled model, and it is difficult to find the optimal ordering. In order to solve this issue approach of reinforcement learning is used in most of the cases. After the standard supervised training procedure the model is further trained using the policy gradient algorithm. In particular, given an input sequence, the decoder computes the reward based on whether the output formed from the processing the input by the model is a well-formed query and if the query will be able to compute the correct results. 5 This reward can be used by the policy gradient algorithm to improve the accuracy and the efficiency of the model.

However, the enhancement in accuracy or performance from the reinforcement learning is quite limited. For example, on a NL2SQL task called WikiSQL, it reports an improvement of only 2% by incorporating reinforcement learning in their model after normal supervised learning.

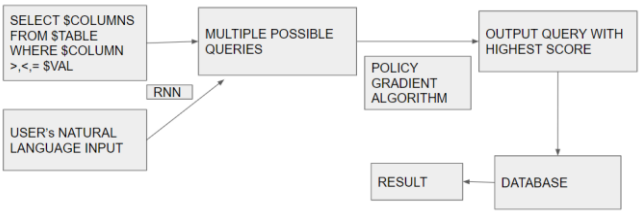

Fig. 8. Query Generation process through SQL Recurrent neural networks and Reinforcement Learning

VI. CONCLUSION

We have demonstrated an interactive natural language query interface for the relational databases. Given a natural language input by the user, our system translated it to the Structured Query Language in a way that aims at constructing the most accurate output SQL query while at the same time minimizing the any further user inputs apart from the natural language. Our project makes use of rnn to process the natural language and in order to increase the efficiency it makes use of the reinforcement learning. When ambiguities are encountered, for each ambiguity, our system generates multiple likely interpretations for the user to choose from, which resolves ambiguities interactively with the user. The model is having a self learning feedback where it has capability of learning through its own mistakes as reported by the user. This makes the model give more accurate answers as the model is used. The query mechanism described in this paper has been implemented, and actual user experience gathered. Using our system, even naive users are able to accomplish logically complex query tasks.

REFERENCES

1. M. Schuster and K. K. Paliwal,(1997), "Bidirectional Recurrent Neural Networks,"IEEE Transactions on Signal Processing, vol. 45, pp. 2673–2681, 1997. (IEEE)
2. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory,"Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
3. A. Gelbukh,(Feb 2006), "Natural Language Processing", Published in Journal of IEEE Xplore.Alex Graves, Abdelrahman Mohamed, Geoffrey Hinton, "Speech Recognition with deep recurrent neural network", Dept. of Computer Science, University of Toronto
4. Fei Li(Univ. of Michigan), H.V. Jagadish (Univ. of Michigan), "NaLIR: An Interactive Natural Language Interface for Querying Relational Database" (Paper Submitted in Springer Journal)
5. Hanane Bais, Mustapha Machkour, Lahcen Koutti (April 2016) - "Querying database using a universal natural language interface based on machine learning" Published in IEEE Journal (May 2016)
6. N. Sangeeth, R. Rejimoan (Oct 2015) - "An intelligent system for information extraction from relational database". Published in IEEE Journal June 2016.
7. Deepthi S, Rejimoan R, Vinod Chandra S - "Maximum Entropy based Natural Language Interface for Relational Database. Published in International Journal of Engineering Research and Technology.
8. Xiaojun Xu (Shanghai Jiao Tong University), Chang Liu, Dawn Song (University of the California, Berkeley) Generating SQL Queries from Natural Language. (UC Berkeley)
9. Alessandra Giordani and Alessandro Moschitti. Translating questions to SQL queries with genera-tive parsers discriminatively reranked. InCOLING, 2012.
10. Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. Learning a neural semantic parser from user feedback. CoRR, abs/1704.08760, 2017.
11. Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables.CoRR, abs/1508.00305, 2015.
12. Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In Advances in Neural Information Processing Systems, pp. 2692–2700, 2015.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)