INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 5, Issue 3)
Available online at: www.ijariit.com

Natural language to SQL

Ankita Makker
ankitamakker@iiitdmj.ac.in
PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, Madhya Pradesh

Gaurav Nayak
gauravnayak@iiitdmj.ac.in
PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, Madhya Pradesh

ABSTRACT

In this research, an intelligent system is designed for users to access the database using natural language. It accepts natural language input and then converts it into an SQL query. Using query language for dealing with databases has always been a professional and complex problem. The system currently handles single sentence natural language inputs and concentrates on MySQL database system. The system accommodates aggregate functions, multiple conditions in WHERE clause, join operations, advanced clauses like ORDER BY, GROUP BY and HAVING. The natural language statement goes through various stages of Natural Language Processing like morphological, lexical, syntactic and semantic analysis resulting in SQL query formation. Intelligent Interface is the need for database applications to enhance efficient interaction between user and DBMS. The research focuses on making the system more dynamic. Improvements have been introduced to the system by incorporating preprocessing of text, named entity recognition, building hierarchical relations, semantic similarity and negation handling using dependency graphs.

Keywords— *Natural Language Processing (NLP), SQL, Semantic similarity, context, Named entity recognition, Dependency graphs*

1. INTRODUCTION

Today, information retrieval technologies are being highly used in various institutions, organizations, companies to manage their information systems and processes. Every Relational Database Management System (RDBMS) uses Structured Query Language (SQL) for querying and maintaining the database.

(a) This makes the service limited to those individuals who are familiar with data query methods. It is a major problem for all those who are not technically knowledgeable in this domain to write queries with the right syntax in SQL.
(b) Accessing the database and manipulating it is a basic necessity, not knowing SQL introduces dependence, leads to reduced productivity.

Artificial Intelligence (AI) and Linguistics, when combined to develop programs, processing and understanding the natural language, becomes possible, thereby helping in its conversion to a query language. Natural Language Processing (NLP) is a component of Artificial Intelligence. It is the ability of a computer program to understand human speech as it is spoken. The development of NLP applications is challenging because the natural language may be easy for people to learn and use but computers traditionally require programming language that is precise, unambiguous and highly structured.

However, human speech is not always precise, it is often ambiguous and the linguistic structure can be different for a sentence with similar meaning. Despite such challenges, NLP can be used to interpret the free text and make it analyzable. NLIDBS are built to optimize search results and produce information with more accuracy. The aim of the system is to reduce the complexity of database querying. The approach used is similar to that introduced by Nandan Sukthankar, 2017

[2] Who made an NLIDB system to incorporate complex queries using table mapping, attribute mapping and clause tagging to generate the resultant query? A similar approach was used by Garima Singh and Arun Solanki, 2016 [1].

The present research extends the existing work further to make the system robust by making it more dynamic. This system incorporates contextual, semantic and dependency information to enhance its performance on unseen entities and negation handling. This system uses Natural Language Processing and a rule-based approach, it does not introduce over-fitting in any way and generalizes well for any database.

2. LITERATURE SURVEY

2.1 Dependency Parsing

There are ways in which the structure of a Natural Language can be described:

(a) POS Tagging: After tokenization makes a prediction of which tag or label most likely applies in the context and tagging them.
(b) Dependency Parsing: It describes the type of syntactic relation that connects the words (child to the head).

Examples can be seen in figure 1 and figure 2.

Filbert Reinaldha and Tricya E. Widagdo, 2014 [3] also used the latter in their work. This system looks at the individual tags with respect to context to identify important words like nouns or verbs and identifies a relationship between those words to understand dependencies.

SpaCy (Library for advanced Natural Language Processing) features a fast and accurate syntactic dependency parser, which is used for analyzing the user query here.

```
list list NOUN NN compound False
students student NOUN NNS nsubj False
not not ADV RB neg True
enrolled enrol VERB VBN ROOT False
in in ADP IN prep True
Physics physics PROPN NNP pobj False
or or CCONJ CC cc True
NLP nlp PROPN NNP conj False
```

Fig. 1: Text, lemma, POS, tag, dependency, stop-word

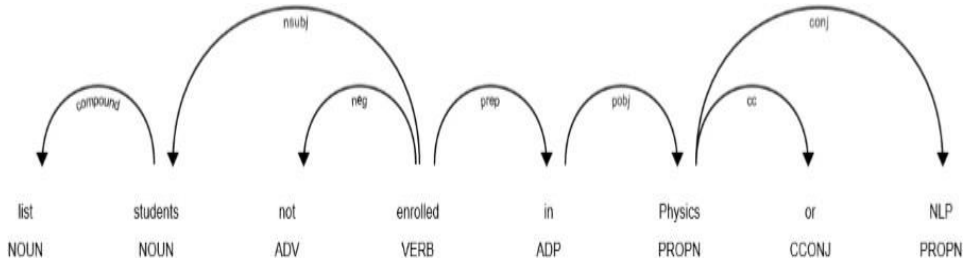

Fig. 2: Dependency Graph for 'List students not enrolled in Physics or NLP'

2.2 Named entity recognition

Phrase Matcher can be used to match large terminology lists which are predefined. Entity recognition helps in labelling contiguous spans of tokens to get an idea of words that the system has not seen before and handle such cases using contextual knowledge. Wordnet synsets (hypernym and hyponym) can be used for a similar purpose. The default model identifies a variety of named and numeric entities, including companies, locations, organizations and products. Arbitrary classes can be added to the entity recognition system and model can be retrained. An example can be seen in figure 3 and figure 4. During the training, examples were batched up using SpaCy's minibatch, and 0.5 was chosen as the droup-out ratio.

```
Jon 0 3 PERSON
```

Fig. 3: "Jon teaches Physics" with the default model

```
Physics 12 19 COURSE
```

Fig. 4: "Jon teaches Physics" with a re-trained model to identify Physics

2.3 Matching

Previously built systems handled sentences which explicitly mention the attribute names as they are in the Database (Ghosh et al., 2014) [4]. Some systems like (Nandan Sukthankar., 2017) [2] handle the problem by a specific substring algorithm (rule-based sequence matching). User can query the system in any way and making it restricted limits the performance of the system. To overcome that and do the match in a more efficient way, a rule-based algorithm that works on sequence matching with semantic similarity (cosine similarity) is used to map the words to attributes and tables. The similarity is determined by comparing word vectors or word embeddings, multi-dimensional meaning representations of a word. Word vectors can be generated using an algorithm like word2vec. Spacy's model that comes with built-in word vectors is used here.

2.4 Table-Attribute Mapping

Queries involving multiple tables and advanced clauses like having or group by and aggregate functions were not incorporated earlier, (Kaur and J, Jan 2016) [5]. nQuery [2] incorporates advanced clauses along with all the simple queries and generalizes

well on different databases. This system also extracts overall database details and uses verb lists, noun lists, implicit hash maps and matching algorithms to map attributes and tables to the words in the sentence. This system partially solves the problem of implicit queries by maintaining hierarchical knowledge from the database schema.

```
nlp = spacy.load('en')
nlp('salary').similarity(nlp('income'))
```

```
0.7829151494399761
```

Fig. 5: Similarity between salary and income

2.5 Negation handling

Most of the systems skip negation handling or simply handle it by inverting conditions on attributes occurring consecutively with negation term in the sentence like in (A.R.FALLE, April 2017) [6]. Another way to do it is by considering n-grams, but all of these approaches fail as the window would be predefined and many a times parameter corresponding to constant is missing in the sentence (implicit cases). This research focused on handling implicit queries, which helped understanding dependencies between various words in the sentence with the negation term. It is important to understand what all parameters are affected with that negation, else results can completely change no matter the amount of processing done to produce it, thereby affecting performance drastically.

One can see, that a rule-based system considering all cases is never guaranteed. The Deep Learning systems that are now coming up are not generalized that well for any database. Research is going on for some larger datasets including complex queries like Spider that can enhance the performance of such strong networks like SQLNet [7]. Another such network is SQLova [8], it is a neural semantic parser translating natural language utterance to SQL query.

But this system is not sensitive to the database information and is not computationally intensive. It generalizes well for any database, mostly covers all cases and is easily implemented.

3. PROPOSED METHODOLOGY

From the above literature survey, analysis of shortcomings of the referred papers and applications along with the future work mentioned was done. The system proposed aims to go beyond the accomplished work. The proposed system is designed to overcome the shortcomings of the existing systems. Input is a natural language sentence, which is then passed through various phases of Natural Language Processing to form the final SQL query (Text pre-processing, Analysis, Table attribute Mapping, Filtering and Query Generation).

3.1 Phase 1. Pre-processing stage

(a) In this stage, the text is converted to a simpler form.

Table 1: Dictionary

Word	Translation
exceed	greater
at most	less equal
at least	greater equal
limit	less

(b) The data is cleaned to remove special characters.
(c) Tokenization and Tagging take place. 'NLTK' package is used for tokenization and Stanford POS Tagger is used for tagging the tokenized array.
(d) Cases are restored, proper nouns are converted to uppercase.
(e) In this stage, dependency parsing of text is done and a long term dependency is maintained in a bottom-up fashion, that helps identify the effect negation terms have on the constants in the conditions of where clause. Negation handling is done by using an inversion array to invert conditions for corresponding negated constants. The constants can be the same as well so to handle dependencies position where that constant occurs plays a critical role.
(f) The improvised system uses SpaCy's retrained Named Entity Recognition model with abstract classes as per the data to identify entity types for implicit data making the system more robust.

3.2 Phase 2. Analyze Tagged Tokens

With the help of clause related data dictionary:
(a) Prepare noun map and verb list from tagged tokens.
(b) The tokens corresponding to various clauses like aggregate, order by, group by, etc. are also mapped with their respective nouns.
(c) The decision whether the natural language statement represents a data retrieval query or DML query taken.

3.3 Phase 3. Table Attribute Mapping

(a) Prepare the table set using noun verb list. This is based on the fact that the table names are either nouns or verbs. The noun map is used to find the attributes that are most important for query generation.
(b) Overall details are fetched from the database through its information schema and implicit maps are constructed.

(c) The table associated with the attribute and the clause tag is stored in an attribute-table map which is used in the final stage of query formation. This is done using the Matching algorithm which first checks for Sequence Matching of substrings (rule-based) after stemming or lemmatizing and then in case an important word got no match, semantic match with a threshold greater than equal to 0.75 is used. For similarity score, cosine similarity is used.

(d) The data obtained during this step i.e. table set and attribute-table map are most likely to be in the final query, which is filtered later.

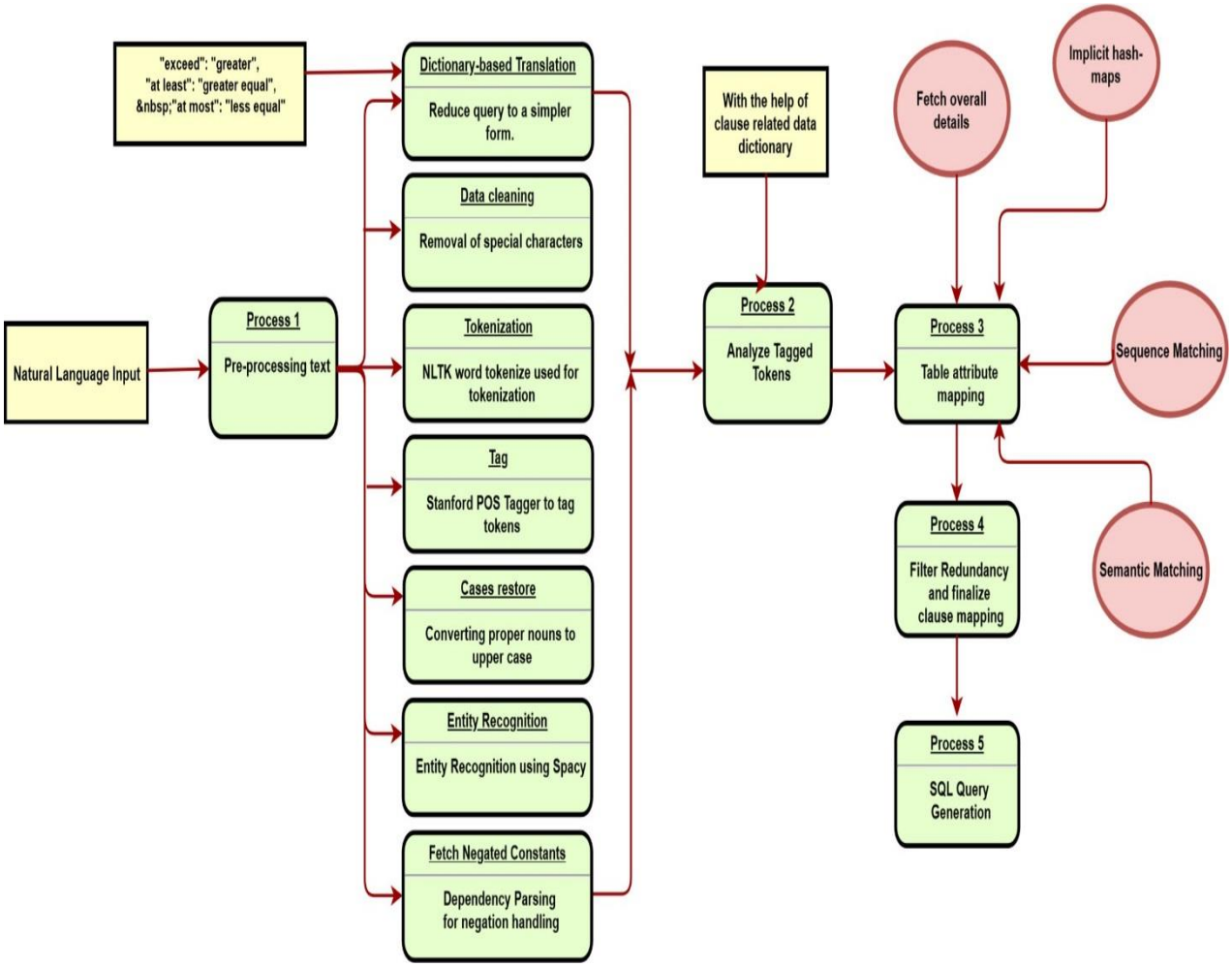

Fig. 6: Data flow diagram

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Fig. 7: Cosine similarity

3.4 Phase 4. Filter redundancy and finalizing clauses

(a) Queries are refined (clauses finalized - Having / Where)

(b) The redundant tables and attributes are removed using some filter algorithms which combine results at different stages. One such combination is a table set after matches and implicit maps.

3.5 Phase 5. SQL Query Generation

(a) The templates used for the query formation will be according to the MySQL syntax.

(b) According to the type of query selected in the second stage of the process (Analyze tagged tokens), the appropriate template is chosen. This process is similar to the one discussed in nQuery[2].

(c) As per final processing, filtered clause objects are substituted. The final check is done for negation constants to modify the query.

Whatever concepts discussed are pipelined to retrieve the final query. They are activated for different cases. One such flow is depicted in Chart.8. In this example, NER and Negation are activated along with the other basic flow of preprocessing, construction of maps and mapping.

In the other example shown in figure 9 Similarity is activated and an implicit query is generated

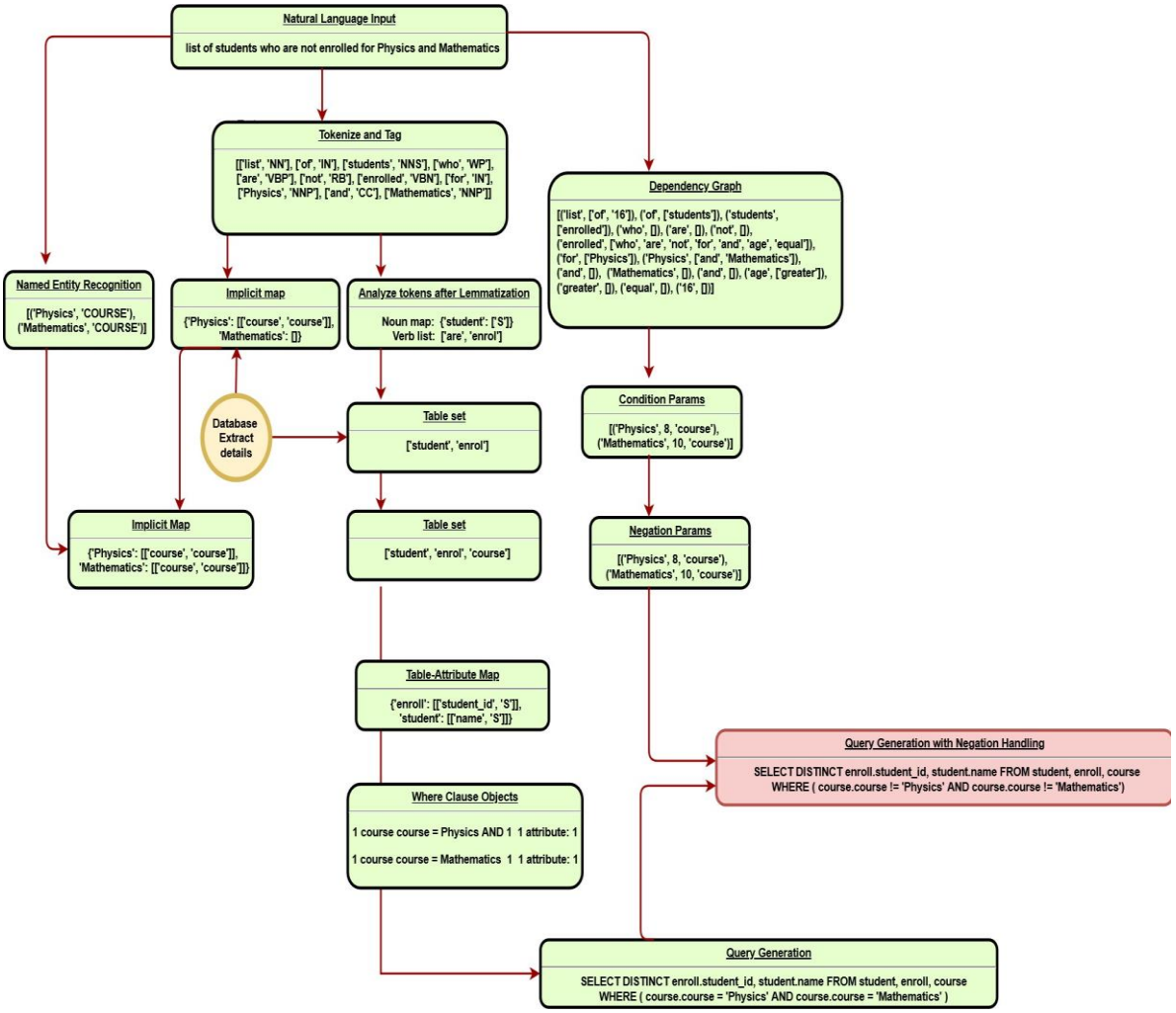

Fig. 8: Basic flow with example (Multi-table query)

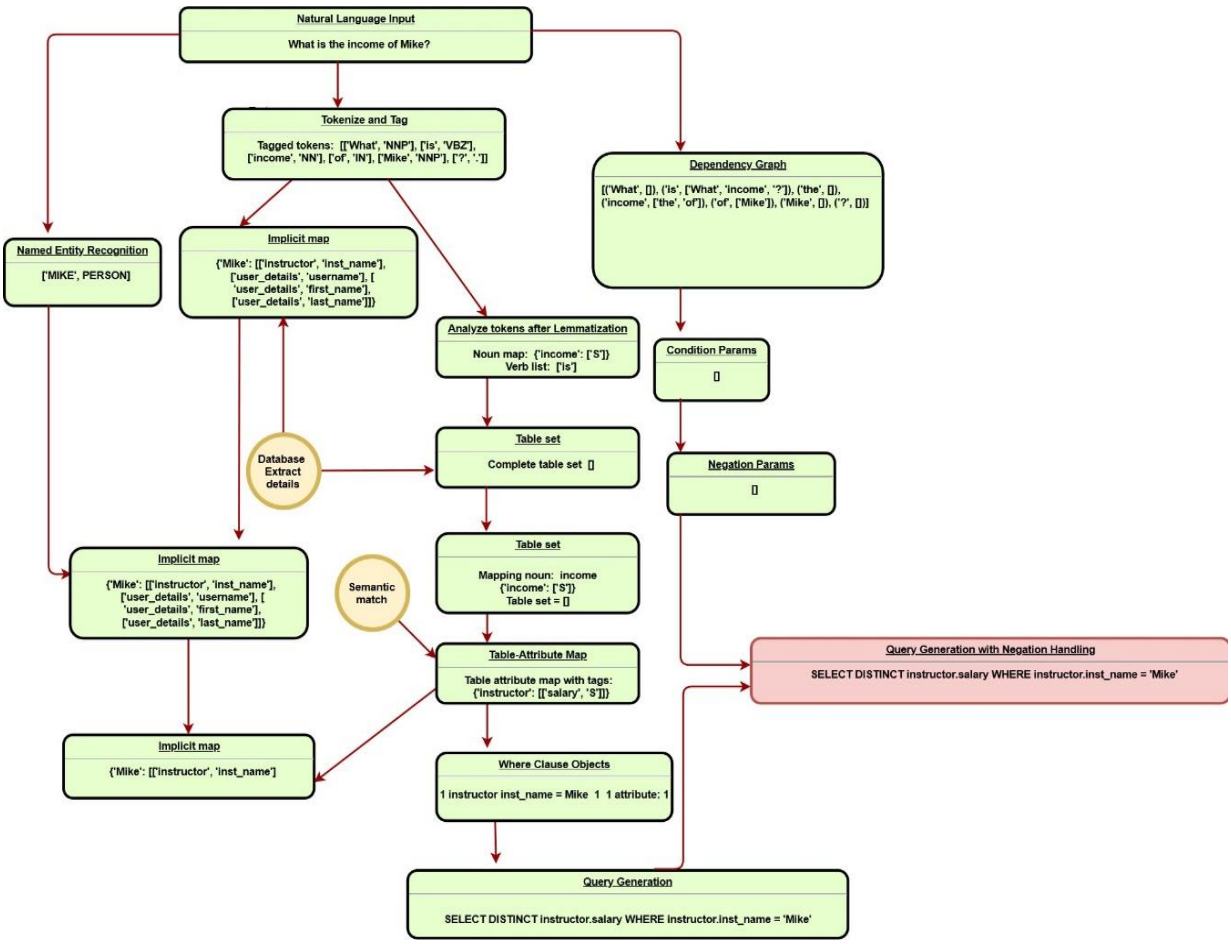

Fig. 9: A step closer to solve the problem of implicit queries (similarity case)

4. RESULTS

Testing of the system is done on a synthesized corpus of natural language statements related to a university database. This system is designed for any complex database, though the input considered is a single length sentence.

The similarity is activated and an implicit query is generated. A qualitative evaluation was done on the queries generated by the proposed methodology. The generated queries were classified as correct (C), partially correct (P) and incorrect (I). Queries are classified as correct when the results they produce are consistent with what was asked by the natural language query.

The queries that are classified as partially correct are those which follows the required template partially and select the required tables. Incorrect queries are those which are not at all consistent with what was asked by the natural language query.

Based on the above mentioned criteria comparison of the results of the proposed methodology with the existing nquery is done. Refer to Table 2, 3 and 4 for the comparative study between nQuery and the new proposed methodology.

Table 2: Comparison between the proposed methodology and nQuery

User Query	nQuery	C/I/P	Natural Language to SQL Conversion	C/I/P
Who teaches Physics?	SELECT * FROM course WHERE course.course = 'Physics'	I	SELECT * FROM instructor, course WHERE course.course = 'Physics'	P
Find the number of hours for which Physics is taught by Mike	NA	I	SELECT COUNT(DISTINCT teaches.hours) FROM teaches, course, instructor WHERE course.course = 'Physics' AND instructor.inst_name = 'Mike'	C
Find instructors with salary exceeding 10000	SELECT DISTINCT instructor.inst_id FROM instructor WHERE instructor.salary = '10000'	I	SELECT DISTINCT instructor.inst_id FROM instructor WHERE instructor.salary > '10000'	C
Find hours for which Mike teach Physics	NA	I	SELECT DISTINCT teaches.hours FROM teaches, course WHERE instructor.inst_name = 'Mike' AND course.course = 'Physics'	C
what is the income of Mike?	NA	I	SELECT DISTINCT instructor.salary WHERE instructor.inst_name = 'Mike'	C
list of students who are enrolled in Mathematics and Physics	SELECT DISTINCT student.name, enroll.student_id FROM student, enroll, course WHERE (.student = 'Mathematics' AND course.course = 'Physics')	I	SELECT DISTINCT enroll.student_id, student.name FROM student, enroll, course WHERE (course.course = 'Mathematics' AND course.course = 'Physics')	C

Table 3: Natural Language queries involving Negation

User Query	nQuery	C/I/P	Natural Language to SQL Conversion	C/I/P
list students not enrolled in Physics or NLP	SELECT DISTINCT student.name, enroll.student_id FROM student, enroll, course WHERE (course.course = 'Physics' OR course.course = 'NLP')	I	SELECT DISTINCT enroll.student_id, student.name FROM student, enroll, course WHERE (course.course != 'Physics' OR course.course != 'NLP')	C
show students with age not less than 5 and enrolled in Physics	SELECT DISTINCT student.name, enroll.student_id FROM student, enroll, course WHERE student.age < '5' AND course.course = 'Physics'	I	SELECT DISTINCT enroll.student_id, student.name FROM student, enroll, course WHERE student.age >= '5' AND course.course = 'Physics'	C
Find instructors with salary not more than 20000	SELECT DISTINCT instructor.inst_id FROM instructor WHERE instructor.salary > '20000'	I	SELECT DISTINCT instructor.inst_id FROM instructor WHERE instructor.salary <= '20000'	C

Table 4: Aggregate functions and multi-table queries

User Query	nQuery	C/I/P	Natural Language to SQL Conversion	C/I/P
list the courses enrolled by Ankita	SELECT DISTINCT course.course_id, course.course FROM course, student WHERE student.name = 'Ankita'	P	SELECT DISTINCT course.course_id, course.course FROM course, student WHERE student.name = 'Ankita'	C
Find the number of hours for which Physics is taught by Mike	SELECT COUNT(DISTINCT teaches.hours) FROM teaches WHERE teaches.hours = 'Physics'	I	SELECT COUNT(DISTINCT teaches.hours) FROM teaches, course, instructor WHERE course.course = 'Physics' AND instructor.inst_name = 'Mike'	C
how many hours is Physics taught?	SELECT * FROM teaches WHERE teaches.hours = 'Physics'	I	SELECT * FROM teaches, instructor, course WHERE teaches.hours = 'Physics'	I
Find the instructor id whose salary is greater than average salary of instructor	SELECT DISTINCT instructor.inst_id FROM instructor WHERE instructor.salary > (SELECT AVG(instructor.salary) FROM instructor)	C	SELECT DISTINCT instructor.inst_id FROM instructor WHERE instructor.salary > (SELECT AVG(instructor.salary) FROM instructor)	C

Significant Improvement is observed. It outperforms all the existing systems when it comes to Negation Handling and Handling implicit cases. The partially correct cases are there because the subject could not be identified that well. Subject object identification limitations lead to reduced accuracy. But maintaining relationships significantly improves query construction as seen in example 1 of Table 2. The system responded correctly for approximately 87% cases for varying queries.

5. CONCLUSION

This paper proposes a development to a Natural Language to SQL system that uses Natural Language Processing in various phases and optimizes results. The fact that the user can not always have exact knowledge about the database entities, attributes or entries. It is about natural expectation to handle cases that are not exactly similar but mean the same thing. Other than semantic similarity, this system also focuses on context as it processes the text to identify entities, it also handles Negation conditions which are not a trivial matter in case of Implicit Queries. An attempt to solve implicit queries is done by maintaining relationships, implicit hash maps and look up dictionaries. The system currently partially solves this problem. Identifying subject object relationship from sentences is easier with dependency graphs but with 'WH' questions, where the subject is not defined explicitly, predicting or identifying what we could be in a sentence is challenging. It can be an attribute from the same table or a referencing table. Research is still ongoing to solve this issue. Also, in a lot of similar systems, Complex queries are not being taken care of. But this system works for Simple as well as complex queries involving natural and inner joins. It works for Aggregate functions in queries, Advanced WHERE clauses, ORDER BY, GROUP BY, HAVING and LIMIT clauses, Basic implicit queries, DML Queries like INSERT, UPDATE and DELETE and negating queries as well. No system till date incorporates such a wide range of queries.

6. LIMITATIONS AND FUTURE WORK

(a) The failed translation is usually caused by a wrong dependency produced by the parser or some linguistic dependency.
(b) The similarity system works on uses a threshold of 0.75, which can lead to wrong translations as well based on the training of the model.
(c) Just how a sentence can be said in many ways meaning the same thing, multiple queries can be generated for it. It is possible that a natural language statement can result in multiple SQL queries. In future, Machine learning can be incorporated to choose the most efficient query.
(d) This system can be made more flexible to handle not Only single sentence natural language input but multiple sentences as well.
(e) A better algorithm to identify subject-object relationships for WH Questions can be developed. Deep Learning can be used as well to handle such implicit Queries in the future.

7. REFERENCES

[1] Garima Singh and Arun Solanki, "An algorithm to transform natural language into SQL queries for relational databases,": International Academy of Ecology and Environmental Sciences, April 2016.
[2] Nandan Sukthankar, Sanket Maharnawar, Pranay Deshmukh "nQuery A Natural Language Statement to SQL Query Generator," Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, April 2017.
[3] Filbert Reinaldha and Tricya E. Widagdo. "Natural language interfaces to the database (nlidb): Question handling and unit conversion, " IEEE, 2014
[4] Prasun Kanti Ghosh, Saparja Dey, and Subhabrata Sengupta. "Automatic sql query formation from natural language query, " International Journal of Computer Applications (0975-8887), International Conference on Microelectronics, Circuits and Systems, (MICRO-2014).
[5] Prabhdeep Kaur and Shruthi J. "Conversion of natural language query to sql," International Journal of Engineering Sciences and Emerging Technologies, Jan 2016.
[6] A.R.FALLE1, S.K.PANHALKAR2. "Knowledge Extraction from Database using Natural Language Processing, " International Research Journal of Engineering and Technology (IRJET), April 2017
[7] Xiaojun Xu. "SQLNet: Geberating structured Queries from Natural Language without Reinforcement Learning," arXiv, 2018
[8] Wonseok Hwang, Jinyeung Yim. "A Comprehensive Exploration on WikiSQL with Table-Aware WordContextualization," arXiv, 2019

BIOGRAPHY

Ankita Makker
Student
PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, Madhya Pradesh, India

Gaurav Nayak
Student
PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur, Madhya Pradesh, India

