Formation of SQL from Natural Language Query using NLP

Uma M
Computer Science and Engineering
SSN College of Engineering
Chennai, India
uma17180@cse.ssn.edu.in

Sneha V
Computer Science and Engineering
SSN College of Engineering
Chennai, India
sneha17160@cse.ssn.edu.in

Sneha G
Computer Science and Engineering
SSN College of Engineering
Chennai, India
sneha17159@cse.ssn.edu.in

Bhuvana J
Computer Science and Engineering
SSN College of Engineering
Chennai, India
bhuvanaj@ssn.edu.in

Bharathi B
Computer Science and Engineering
SSN College of Engineering
Chennai, India
bharathib@ssn.edu.in

***Abstract*—Today, everyone has their own personal devices that connects to the internet. Every user tries to get the information that they require through internet. Most of the information is in the form of a database. A user who wants to access a database but having limited or no knowledge of database languages faces a challenging and difficult situation. Hence, there is a need for a system that enables the users to access the information in the database. This paper aims to develop such a system using NLP by giving structured natural language question as input and receiving SQL query as the output, to access the related information from the railways reservation database with ease. The steps involved in this process are tokenization, lemmatization, parts of speech tagging, parsing and mapping. The dataset used for the proposed system has a set of 2880 structured natural language queries on train fare and seats available. We have achieved 98.89 per cent accuracy. The paper would give an overall view of the usage of Natural Language Processing (NLP) and use of regular expressions to map the query in English language to SQL.**

***Index Terms*—Natural Language Processing (NLP), Structured Query Language (SQL), Tokenization, POS tagging, Chunking, Parsing, Regular Expression**

I. Introduction

In this fast technologically advancing world, it has become important for the humans to interact with computers to provide assistance in many fields like medicine, education, space research, etc. Retrieval of the required information from the database is a tedious process. In order to extract information from the database, the user must have a prior knowledge on Database Management System (DBMS), which is a software developed to store and manipulate the data in a database. Hence a non-technical user faces difficulty in extracting the data. To find a solution for such a problem and facilitate human interaction with computers, Natural Language Processing(NLP) techniques are used. Natural Language Processing has applications in various sectors like tourism, where a tourist can get information about the famous tourist spots in a particular city, the hotels available, best restaurants nearby etc. Another important application of NLP is chatbot (Chat Robot) that can be used for voice or textual interactions. Our work focuses on Railway Reservation System, where a user can enquire about the trains that are available from a source to destination, the fare of a ticket in various classes. Objective of this paper is to convert a natural language query into a SQL to simplify data extraction. The use of natural language interface to access database was topic of research since the beginning of NLP and is still an interesting problem.

II. Literary Study

In 1972, [1] W.A. Woods developed a system that provided a search interface for the database system that stored information about the rock samples that were brought from the moon for research. This system used two databases, the chemical analyses and the literature references. This system used Augmented Transit Network (ATN) parser and semantics of Woods. This system was demonstrated informally at Second Annual Lunar Science conference in 1971. Lifer/Ladder system (1978) [2] was one of the good search interface techniques (i.e. NLP system) which used semantic grammar for parsing the input query and the query generated was given as input on a distributed database system. This system supports single table queries and simple join queries in case of multiple tables. Akshay et al. [7] proposed a system which provides a search interface for the users to pose questions in their natural language. The primary goal of this system is to generate a database language query from a NL query. This system includes an additional feature of eliminating spelling errors from user queries and used Word Pair Mining Technique for the same. Then the query in English is mapped to an equivalent SQL query. Prasun Kanti et al. [11] has proposed a system for interfacing a college database that transforms English query to SQL using semantic grammar. The system goes through the morphological, syntactic, semantic phases. The user may ask the question in speech format which is then converted to text using Scripting Language for

Authorized licensed use limited to: PES University Bengaluru. Downloaded on June 16,2025 at 05:33:54 UTC from IEEE Xplore. Restrictions apply.

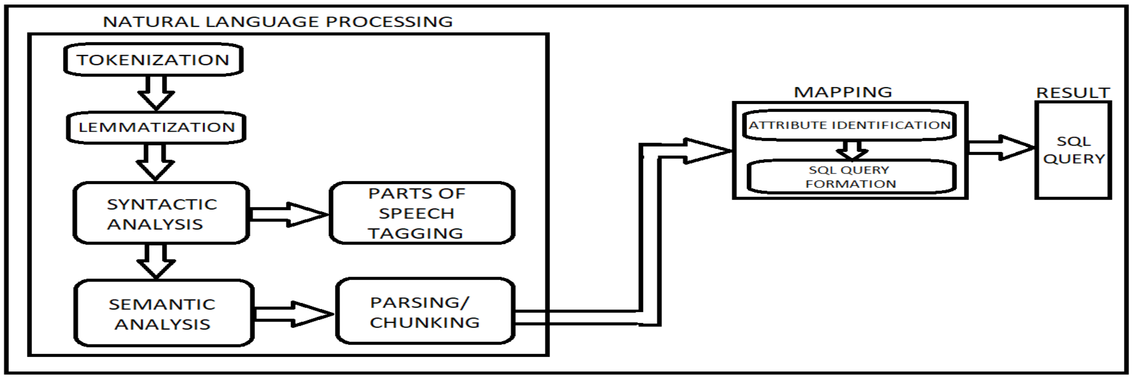


Fig. 1: Proposed System Model

Android (SL4A). The natural language query is then parsed using parser. A data dictionary stores all the attributes and tables of the database. The attribute identifier then finds out the attributes that are present in the natural language query. With the identified attributes, SQL query is generated [9], [10]. K. Javubar et al. [3] has proposed a user-friendly interface for accessing data from various web sources such as Facebook, Twitter, etc. The architectural layout consists of tokenizing, stemming, parsing and mapping stages. The input natural language query initially undergoes morphological analysis then semantic analysis which is followed by a mapping phase. The three main keywords SELECT, FROM and TO are looked for in the input query. Once these are found, the SQL query is formed [8].

III. Proposed System

Retrieving the required information from a database is quite difficult for any common man and requires a lot of effort which needs the knowledge of the database structure. DBMS is incapable of dealing with queries framed in any other languages other than the standard database languages. So to make the retrieval more effortless and interactive for naive user, our proposed work provides a facility through which a user is free to pose a query in English, which will be processed by several modules to form an equivalent SQL query.

A. *Overview of Query Formation*

User submits an English query in the text form which is then sent into several natural language processing (NLP) modules. This NLP phase is followed by a mapping phase in which the attributes are detected in the English query, mapped to form the final SQL query and may then be fed into the database to retrieve the required information and provide it to the user. The overview of our proposed work is elucidated in Fig. 1. As of now, the proposed work focuses on generation of an equivalent SQL query from a natural language question in English. Once the SQL query is generated the retrieval of data from DB will be an easy task.

B. *Algorithm*

```
Algorithm: Formation of SQL Query from Natural Language Query
Input: Natural Language query in English text.
Output: SQL query
1) Tokenize the input into list of words
2) Lemmatize the list of words
3) Perform POS tagging
4) Parsed_ sentence = Parse using regular expressions
5) If table. attribute $\in$ Parsed_ sentence
a) Extract them
b) Call SQLmap()
```

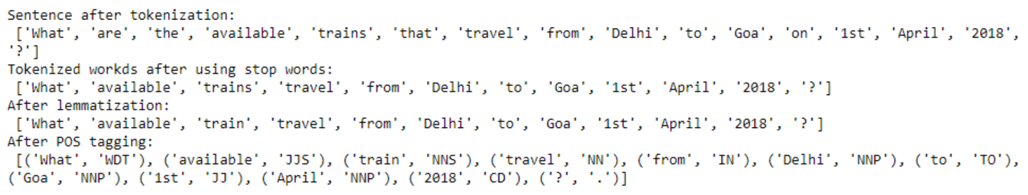
C. *Proposed System Model*

Our proposed system consists of several modules that are used to extract key words alone and leave out the redundant data. This is critical because presence of redundant data will certainly decrease the overall performance of the system. Input data initially goes through an NLP phase followed by a mapping phase. The NLP phase consists of processes such as tokenization, lemmatization, Parts Of Speech tagging (POS tagging) and parsing. The mapping phase identifies the attributes in the processed input and finally the SQL query is formed from the key information that was obtained [4], [6]. The detailed workflow of our proposed work is given in Fig.1.

1) Tokenization: It is the first step that is used to break a sentence into smaller meaningful tokens in most cases these are words. In the proposed system, we applied tokenization as soon as the text input is received from the user and the tokens obtained are stored in the form of a list. We have used *word_tokenize* module of *nltk.tokenize* library in Python.

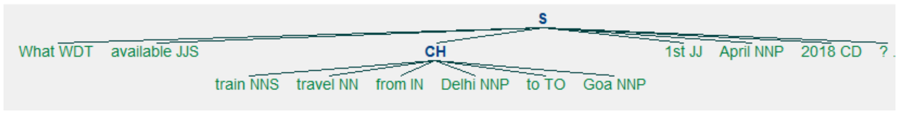
2) Lemmatization: This process is similar to stemming where the root words or lemma of each of these tokens are obtained from the output of the previous step and are stored in another list. Lemmatization is chosen over stemming because the process of stemming does not always prove to be accurate since it removes simply the prefix or suffix of a word. Whereas in lemmatization, the roots are matched with its lemmas contained in a dictionary and hence more accurate results were obtained.

Authorized licensed use limited to: PES University Bengaluru. Downloaded on June 16,2025 at 05:33:54 UTC from IEEE Xplore. Restrictions apply.

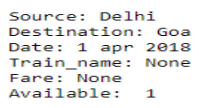


```
Sentence after tokenization:
 ['What', 'are', 'the', 'available', 'trains', 'that', 'travel', 'from', 'Delhi', 'to', 'Goa', 'on', '1st', 'April', '2018',
'?']
Tokenized workds after using stop words:
 ['What', 'available', 'trains', 'travel', 'from', 'Delhi', 'to', 'Goa', '1st', 'April', '2018', '?']
After lemmatization:
 ['What', 'available', 'train', 'travel', 'from', 'Delhi', 'to', 'Goa', '1st', 'April', '2018', '?']
After POS tagging:
 [('What', 'WDT'), ('available', 'JJS'), ('train', 'NNS'), ('travel', 'NN'), ('from', 'IN'), ('Delhi', 'NNP'), ('to', 'TO'),
('Goa', 'NNP'), ('1st', 'JJ'), ('April', 'NNP'), ('2018', 'CD'), ('?', '.')]
```

(a) Tokenizing, lemmatizing and parts of speech tagging for question type 1

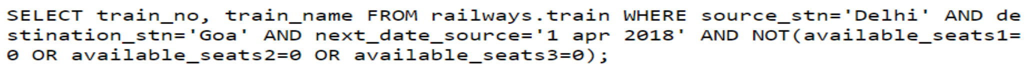


(b) Parse tree for question type 1



```
Source: Delhi
Destination: Goa
Date: 1 apr 2018
Train_name: None
Fare: None
Available:  1
```

(c) Values for question type 1



```
SELECT train_no, train_name FROM railways.train WHERE source_stn='Delhi' AND de
stination_stn='Goa' AND next_date_source='1 apr 2018' AND NOT(available_seats1=
0 OR available_seats2=0 OR available_seats3=0);
```

(d) Structured Query Language query for question type 1

Fig. 2: Stages of proposed System

3) Syntatical Analysis: In syntactic analysis, each of the lemmatized tokens are analyzed and according to their context of appearance, each token is tagged with a POS. Here, each word and its tag are packed into a tuple and a list of all such tuples is obtained. In the proposed system, the *Stanford POS Tagger* is used for POS tagging. This tagger is preferred over the POS tagger present in the *NLTK* package as it provides more accurate tagging.

4) Semantic Analysis: In semantic analysis, we try to make sense of the tokens so that the system could proceed with the SQL query formation. This is achieved by the process of parsing (or chunking). In the proposed system, the *RegExpParser()* (regular expression parser) is used for parsing the POS tagged input data. This parser chunks the data based on a regular expression. In our work, a regular expression is framed such that a phrase has the source and destination information is classified into a separate chunk and are extracted by means of a rule-based paradigm. The regular expression that is used in the project is:

$$CH : \{(< VB.? > | < NN.? > | < JJ.? >) < NN > * < RB.? > *(< IN > | < TO >)? < NNP > < NN.? > * < NN > * < VB.? >?(< IN > | < TO >)? < NNP >< NNS >? < VB.? >?\}$$

POS tags expansion: <> Enclosing brackets for all POS tags; VB *Verb*; NN *Noun*; JJ *Adjective*; RB *Adverb*; IN *Preposition*; TO *to*; NNP *Proper Noun*; NNS *Noun* (Plural form). Regular expression syntax overview: . any character; ? occurrence of 0 or 1 times of the preceding character; * occurrence of 0 or more times of the preceding character ; || OR condition; () Group; CH Name given to the string (or chunk) that matches the above regular expression [5]. According to this regular expression, we detect a pattern such that it contains the source and destination in an extractable form. The pattern is such that the beginning of the chunk is either a verb or a noun. The condition includes a check for noun tag because sometimes the POS tagger tags certain verbs as nouns. Followed by this there may be a noun (optional denoted by *) or adverb (optional denoted by *). This is followed by either a from (POS Tag IN) keyword or to keyword (POS tag TO). When these keywords appear, it is predicted that the user would give the source information immediately after from and the destination information immediately after to keyword. Hence the regular expression specifies a check for a proper noun following from or to (since all locations are proper nouns). The proper noun may be followed by a noun(optional) or verb (optional) to include information such as station or junction (which are nouns). Once again there is the same pattern of from or to followed by location followed by optional noun or verb. If the first pattern detected the source, the second pattern would detect the destination and vice-versa. Finally, this regular expression is fed into the RegExpParser() which looks for this pattern in the POS tagged input data. When the pattern is detected, it is classified as a separate chunk named CH which appears as a subtree in the tree diagram of the parsed data.

5) Attribute Identification: An attribute is a column in the database table. A rule-base paradigm is followed for extracting the attributes from the parsed data. a) Source - Destination Identifier: For finding the source and destination (if any), we make use of the list that was created in the last step. For each element in the list, if the element is a preposition (from or to), we extract the immediately following proper noun (location) as source (if from is encountered) or destination (if to is encountered). In some cases, the user may not use a preposition to specify the location details. In such cases, we check if each list element is a verb. If it is a verb, we make the system understand the meaning of the verb using *Wordnet* module. Initially we create a list of all possible verbs that the user may use and then we compile the list of all possible synonyms of those words. These synonyms can be

Authorized licensed use limited to: PES University Bengaluru. Downloaded on June 16,2025 at 05:33:54 UTC from IEEE Xplore. Restrictions apply.

obtained by using the *synsets()* function of Wordnet. For each verb encountered, we check if the word lies in the synonyms of any of the predefined words we created. If it matches, then the proper noun following that verb is extracted as source or destination accordingly. If the verb encountered is similar to starting or departing, the proper noun following it will be the source. If the verb is similar to reaching, stopping or arriving, the proper noun following it will be the destination. If there is no source or destination or if the system could not detect any source and destination, then the variables to the left of the calling function will have None value.

b) Date Identifier: For extracting date (if any), we make use of the datetime module. As the CH subtree of the parsed tree diagram contains only the source and destination, the date will be present outside the CH chunk. We look for the tag CD (Cardinal Digit) for any number. Regular expressions are used to find certain patterns of date. The date can be of any format where the month is at the middle such as DD/MM/YYYY. Apart from this, the user may give the date in formats like [MONTH][DAY][YEAR] or [DAY][MONTH][YEAR] like December 29th, 2018 or 29th December 2018. If element in the list of tuples is 29 or 29th, it is tagged as CD, NN or JJ. If it is detected, then the tag of the tuple at index before and after, or two index positions after the tuple in which the day (e.g. 29) is found are compared to find the date. If th, rd, or st is present in the word, it is replaced with an empty string. Using regular expression *(jan+)|(feb+)|(mar+)|(apr+)|(may+)|(jun+)|(jul+)|(aug+)|(sep+)|(oct+)|(nov+)|(dec+)*, we can detect the month in the list. Then datetime() is used to separately get the day, month and year in a regularized format as it had a formatted datetime object returned. If no date is detected, nothing is returned to the calling function and hence the values of the variable at the left of the calling function has None value.

c) Fare Identifier: The proposed system can handle two types of queries: one in which the user wants to know the train number and train name and the other in which the user wants to know the fare. To detect if the user wants to know about the fare, we search the set of lemmatized words for the keyword fare. If it is present, we search the succeeding and preceding words for the class name of which the fare is needed. The class names maybe class1, class2 or class3. Regular expressions are used for this purpose. The regular expression *class[.][123]* is used to find out the class name. Once the class name is identified, its corresponding fare such as fare1, fare2 or fare3 is returned. If the user has not posed any fare regarding questions, then the variable at the left of fare_identifier() will have None value.

d) Train Name Identifier: In some cases, the users would give the train name with or without the source destination and date information. To detect the train name(s) (if any) is present in the input data, we use the regular expression *'(express)|(mail)|(passenger)'*. According to this regular expression, the input data is searched for the words express, mail and passenger. If they are found, their preceding word along with the matched word is taken as the train name and it is returned. If no such train name could be found, then the calling function had None value.

e) Available Identifier: When the user asks if a train is available, the system needs to display only the trains which have seats available for booking in them. Hence the system must check if the trains that are to be displayed have at least one empty seat. For this, the set of input data is searched for the keyword available. If it is found, an additional part is to be added to the SQL query being formed. This part makes sure that there is at least one seat in either class1 or class2 or class3. If the available keyword is not found, then the calling function will have None value. The algorithm for SQL query formation is given below.

Algorithm: SQLmap()

```
query= "SELECT"
if (value = Fare_Identifier())
query = query + value
else
query = query + train_no + train_name
query = query + "FROM railways.train WHERE"
if (value= Train_Name_Identifier ())
query = query + "train_name=" + value
if (source, destination = Source_Destination_Identifier())
query = query + "source_stn=" + source +
"destination_stn=" + destination
if (date= Date_Identifier())
query = query + "next_date_source" + date
```

IV. EXPERIMENTS AND RESULTS

We have taken the domain of railway reservation for our query translation. We have considered only a single table *railways.train* for which the SQL query would be produced. There are 24 attributes in the table, they are: *train_no, train_name, source_stn, destination_stn, arrival_time, departure_time, available_days, next_date_source, next_date_destination, total_seats1, total_seats2, total_seats3, booked_seats1, booked_seats2, booked_seats3, waiting_seats1, waiting_seats2, waiting_seats3, available_seats1, available_seats2, available_seats3, fare1, fare2, fare3*. We have implemented the proposed system in Python3 and using the following packages: *nltk, re, datetime, nltk.tokenize, nltk.tag, nltk.stem, nltk.corpus*. The questions that are given as input to the proposed systems are of two types: 1. What are the trains that are available from source to destination on a given date? 2. What is the fare of a specific class of the train travelling from source to destination on a given date? The dataset containing 2880 NL queries was formed using the above two formats of questions. The dataset was collected from friends and family by asking them to pose the same question in their own words. Let us consider an example to illustrate the working of the algorithm. Question: *What are the available trains that travel from Delhi to Goa on 1st April 2019?* The input sentence is split into words and stored in a list, i.e. tokenized, followed by removing unnecessary words by using stop_words which is a list containing words that are of

Authorized licensed use limited to: PES University Bengaluru. Downloaded on June 16,2025 at 05:33:54 UTC from IEEE Xplore. Restrictions apply.

not much importance. Then the list is lemmatized. Next, the words are tagged using *StanfordPOSTagger()*. The output of the above steps is shown in Fig. 2(a). During the POS tagging, sometimes the words get tagged incorrectly. Hence some of the tags are changed to keep it regularized. Then, the list is parsed using the *RegexParser()* to which regular expression is given as argument. The regular expression is in such a way that it chunks the list so that the chunked part contains the source and destination. The regular expression is formed by observing the POS tagging of various sentences. The parse tree after parsing is given in Fig. 2(b). The above step is followed by mapping. There are five modules/functions in the program code to identify source-destination, fare, date, train name and the word available from the list. If any of them is not available in the list, their respective values would be None. For this example, the values are given as shown in Fig. 2(c). The values in Fig.2(c) are mapped to form the query in SQL. If source, destination, available and date are not None, then the SQL query would display the train number and train name of trains that have seats available in any of the three classes of the train travelling from that source to destination on the given date. The query for the given example is shown in the Fig. 2(d). Total number of queries as test cases are 2880, correctly generated queries are 2848 and incorrectly generated queries are 32.

TABLE I: PERFORMANCE METRICS

Metric	**Result obtained**
Precision	0.5
Recall	0.494
F1 Score	0.497
Accuracy	0.9889

V. CONCLUSION

Although several methodologies are employed to extract information from a database, Natural Language Processing has set a new standard in doing the same. This work presents a clear picture on the steps that are involved in NLP. Various processes like tokenization, lemmatization, syntactic and semantic analysis are carried out to generate an equivalent SQL query from a natural language query. We have obtained an accuracy of 98.89 per cent.

Following are the future improvements that can be incorporated; The input received can be of audio form, which can be converted into textual format; The SQL query could be of greater complexity; The database could be larger in terms of attributes and tuples. Also, there could be multiple tables of related data which can be accessed using JOIN keyword; It could be used to create chatbots for various sectors which handle large databases and can help users to access them with greater ease; The output could be converted as a sentence then into audio format to make the system more interactive; This work can also be extended to other languages. To conclude, NLP is a boon to any ordinary person having no knowledge on database management. In short, NLP proves to be a promising tool for future advancements in numerous fields like medicine, business, and education so on.

REFERENCES

[1] Woods, William, (1972) "The lunar sciences natural language information system," BBN report, Bolt Beranek and Newman.

[2] Hendrix, Gary G, Sacerdoti, Earl D and Sagalowicz, Daniel and Slocum, Jonathan, (1978) "Developing a natural language interface to complex data," ACM Transactions on Database Systems (TODS), vol. 3, Issue 2, pp. 105–147.

[3] Sathick, K Javubar and Jaya, A, (2015) "Natural language to SQL generation for semantic knowledge extraction in social web sources," Indian Journal of Science and Technology, vol. 8, Issue 1, pp. 1–10.

[4] Singh, Garima and Solanki, Arun, (2016) "An algorithm to transform natural language into SQL queries for relational databases," Selforganizology, Directory of Open Access Journals, vol. 3, Issue 3, pp. 100–116.

[5] Huang, Bei-Bei, Zhang, Guigang et al., (2008) "A natural language database interface based on a probabilistic context free grammar," IEEE International workshop on Semantic Computing and Systems, pp. 155–162.

[6] Rao, Gauri, Agarwal, Chanchal, Chaudhry, Snehal, et al., (2010) "Natural language query processing using semantic grammar," International journal on computer science and engineering, vol. 2, Issue 2, pp. 219–223.

[7] Satav, Akshay G, Ausekar, Archana B and Bihani, et al., (2014) "A Proposed Natural Language Query Processing System," International Journal of Science and Applied Information Technology, vol. 3, Issue 2, pp. 219–223.

[8] Iftikhar, Anum, Iftikhar, Erum, Mehmood and Muhammad Khalid, (2016) "Domain specific query generation from natural language text," IEEE Sixth International Conference on Innovative Computing Technology (INTECH), pp. 502–506.

[9] El-Mouadib, Faraj A, Zubi, Zakaria S, Almagrous, Ahmed A, El-Feghi and Irdess S, (2009) "Generic interactive natural language interface to databases (GINLIDB)," International journal of computers, vol. 3, Issue 3, pp. 301–310.

[10] Bhadgale, Anil M, Gavas, Sanhita R, Patil, Meghana M and Pinki, R, (2013) "Natural language to SQL conversion system," International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR), Vol. 3, Issue 2, pp. 161–166.

[11] Ghosh, Prasun Kanti, Dey, Saparja, Sengupta and Subhabrata (2014) "Automatic sql query formation from natural language query," International Journal of Computer Applications (0975-8887), International Conference on Microelectronics, Circuits and Systems (MICRO-2014)

Authorized licensed use limited to: PES University Bengaluru. Downloaded on June 16,2025 at 05:33:54 UTC from IEEE Xplore. Restrictions apply.