Dialogue based Question Answering System in Telugu

Rami Reddy Nandi Reddy
Dept. of Comp. Sc. & Engg,
Jadavpur University,
Kolkata, India
nramireddy@gmail.com

Sivaji Bandyopadhyay
Dept. of Comp. Sc. & Engg,
Jadavpur University,
Kolkata, India
sivaji_cse_ju@yahoo.com

Abstract

A dialogue based Question Answering (QA) system for Railway information in Telugu has been described. Telugu is an important language in India belonging to the Dravidian family. The main component of our QA system is the Dialogue Manager (DM), to handle the dialogues between user and system. It is necessary in generating dialogue for clarifying partially understood questions, resolving Anaphora and Co-reference problems. Besides, different modules have been developed for processing the query and its translation into formal database query language statement(s). Based on the result from the database, a natural language answer is generated. The empirical results obtained on the current system are encouraging. Testing with a set of questions in Railway domain, the QA system showed 96.34% of precision and 83.96% of dialogue success rate. Such a question answering system can be effectively utilized when integrated with a speech input and speech output system.

1 Introduction

Ever since Question Answering (QA) emerged as an active research field, the community has slowly diversified question types, increased question complexity, and refined evaluation metrics, as reflected by the TREC (Text Retrieval Conference) QA track (Voorhees, 2004). Several QA systems have responded to these changes in the nature of the QA task by incorporating various knowledge resources (Hovy et al., 2002), handling of additional types of questions tapping into external data sources such as web, encyclopedia, and databases in order to find the answer candidates, which may then be located in the specific corpus being searched (Xu et al., 2003).

The most popular classes of technique for QA are open-domain and restricted-domain (Diekema et al., 2004, Doan-Nguyen et al., 2004). These two domains use thesauri and lexicons in classifying documents and categorizing the questions. Open domain question answering deals with questions about nearly everything and can only rely on general ontology. It has become a very active research area over the past few years. On the other hand, Restricted-domain question answering (RDQA) deals with questions under a specific domain. If we create such a RDQA interface for structured e.g. relational database, we call it as Natural language interface to database system (NLIDB) (Androutsopoulos et al., 1995), where it allows the user to access the information stored in database by typing requests expressed in some natural language. RDQA has a long history, beginning with systems working over databases (e.g., BASEBALL (Green et al., 1961), and LUNAR (woods et al., 1972)).

In practice, current QAs can only understand limited subsets of natural language. Therefore, some training is still needed to teach the end-user what kinds of questions the system can or cannot understand. There are kinds of questions (e.g. questions involving negation, or quantification) that can be easily expressed in natural language, but that seem difficult (or at least tedious) to express using graphical or form based interfaces. Anaphoric and elliptical expressions are also handled by the QA systems. In recent years a large part of the research in QAs has been devoted to portability, i.e., to the design of QAs that can be used in different knowledge domains

(Knowledge domain portability), with different underlying Database Management System (DBMS) (DBMS portability), or even with different natural languages (Natural language portability). There is a growing body of research on integrating speech recognition, robust interpretation with the goal being to implement systems that engage users in spoken dialogue to help them perform certain tasks. We expect that this line of research will have a significant influence on future QAs, giving rise to systems that will allow users to access databases by spoken dialogue, in situations for which graphic and form-based interfaces are difficult to use.

A practical question answering system in restricted domain (Hoojung et al., 2004) and our system handles user questions similarly. However, our system extracts the information from a relational database. Moreover, our system keeps track of user dialogue and handles clarifications, elaborations and confirmations needed from the user with respect to the query. Along with it returns natural language answer in user-friendly format.

ARISE (Automatic Railway Information System for Europe) is a spoken dialogue system to provide train timetable information over the phone. Prototypes have been developed in four languages: Dutch, French, English, and Italian. ARISE uses a mixed initiative Dialogue Manager (DM). A mix of implicit and explicit confirmation is used, based on how confident the system is in deciding whether an item has been correctly understood.

We relate this paper as an experiment for designing a keyword based QA system for a huge domain (i.e. for Railways), which aims at replying users questions in their native language (Telugu). The system generates SQL query out of the natural language question, executes the SQL query over a relational database and then provide the answer. Dialogue Manager (DM) is maintained to generate dialogues with user and to handle the anaphoric and elliptical expression in our query. This system is implemented on a relatively restricted domain that includes a number of aspects of railway information system (Arrival/Departure time, Fare between for particular stations, Trains between important stations etc.). The precision of the information extraction stage is essential to the success of a QA system, because it places an upper bound on the precision of the entire system.

The empirical results obtained on the current system are encouraging. Testing with a set of questions in Railway domain, the QA system showed 96.34% of precision and 83.96% of dialogue success rate.

Section 2 deals with the System Architecture of the QA system. Section 3 details about the QA system design in the Railway information domain using the Keyword based approach. The evaluation has been carried out in Section 4. Section 5 concludes with some directions for future work.

2 System Architecture

In this keyword based approach the input query statement is analyzed by the query analyzer, which uses domain ontology stored as knowledge base, generating tokens and keywords. The appropriate query frame is selected based on the keywords and the tokens in the query statement. Each query frame is associated with a SQL generation procedure. The appropriate SQL statement(s) is generated using the tokens retrieved from the input query.

The QA system architecture is shown in Figure 1. The Dialogue Manager keeps track of the elliptical queries from the user that constitute the dialogue and helps in the SQL generation procedure using dialogue history (Flycht-Erikson et al., 2000), which contains information about previous tokens and their types as well as other dialogue information like answers retrieved by the current SQL statements and the answers for previous queries in the dialogue. The SQL statements used to retrieve the correct answer from the database. Based on the result of the DBMS, a natural language answer is generated. This answer is forwarded to the DM for onward transmission to the user.

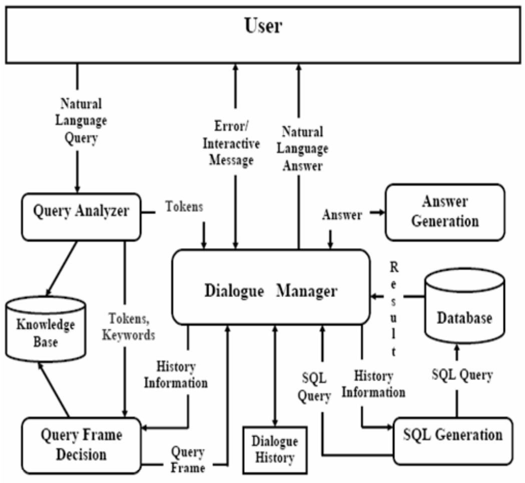


Figure 1. QA System Architecture

If the system cannot decide on the query frame by using the keywords extracted from the input query, the system enters into a dialogue with the user through the DM. During SQL generation if it is detected that more information is needed from the user to generate the SQL statement then an interactive message is sent to the user through the DM. The user will then send the needed information to the system. If user could not provide correct information then DM sends an error message to the user indicating the error in the user query. In case, the SQL statement generates a null response from the database the DM will send a cooperative message depending on the user query.

3 Design of Railway Information System

The most important issue in the design of the Railway information system is the design of the Railway database and the Knowledge base. These are detailed in Sections 3.1 & 3.2 respectively. The different components of the dialogue based QA system, i.e., Query Analyzer, Query Frame Decision, Dialogue Manager, SQL Generation and Answer Generation sub systems are described in subsequent sections.

3.1 Railway Database Management

The system as a whole is engaged in data access, and is a hybrid system with subsystem to analyze the natural language query and formal query language SQL, and a data retrieval and database management system. The *database* is structured and contains the information to provide the railway information service. For example in a Railway information system, database contains information about the arrival/departure time of trains, their fares and their running information etc. The aim of database management is to describe the information, in order to offer the service.

For our purposes the relational model has important advantage: The relational model stresses on data independence. This means that the user and front-end programs are effectively isolated from the actual database organization.

The main tables used here are schedule table for each train, fare tables for special trains like Rajdhani, Shatabdi etc. that have a different fare structure, Route tables for each route and tables that include train running frequency details etc. Some temporal tables are maintained in order to check the status of the railway ticket (which is known as checking the Passenger Name Record or PNR status of the ticket) and reservation availability information of a particular train.

3.2 Design of the Knowledge Base

The system maintains a knowledge base of the domain to facilitate question answering. For a system operating on a restricted domain this is quite obvious since it will greatly improve the disambiguation and parsing.

The words that occur in the database query for Railway information system includes words describing train name, station name, reservation class, and date and/or period of journey or keywords that specify the topic of the query. Hence we stored a domain dependent ontology in the knowledge base.

Knowledge base, which contains tables for train name, station name and alias tables for train name and station name. We have stored possible Telugu inflections (కి (*ke* [to]), కు (*ku* [to]), లో (*loo* [in]), తుంది (*tundi* [ing]), వి (*vi* [have]) etc. for ex: గుంటూరుకు (*gunturku* [to Guntur])), which can be used in *morphological analysis* of input query**.** We have considered possible postpositions like నుండి (*nundi* [from]), నుంచి (*nunchi* [from] etc. (For ex: న్యూఢిల్లీ నుండి (*newdelhi nundi* [from New Delhi])), which can be used to identify the source station in the input query and route words like దగ్గర (*daggara* [near])**,** ద్వారా (*dwara* [through])**,** గుండా (*gunda* [through])**,** వద్ద (*vadda* [at])**,** మీదుగా (*meedugaa* [via]) etc. (For ex: గయా మీదుగా (*gaya meedugaa* [via Gaya])), which can be used to identify the route station of the journey. We kept a list of keywords in a table in order to identify the proper query frame.

3.3 Query Analyzer

During query analysis, Morphological analysis of the input query statement is carried out to identify the root words / terms. Analyzing the whole input query, the system identifies several tokens such as Train name, Station name, Reservation class, date and period of the day etc. and a set of keywords.

The query analyzer consults the domain-dependent ontology i.e. knowledge base for recognizing these tokens and keywords. It may happen that some words/terms may not found in the knowledge base. Those words do not contain any semantic information and are simply discarded.

For example: If our input query is ఎప్పుడు ఫలక్ నుమా ఎక్స్‌ప్రెస్ గుంటూరుకు వెళ్తుంది (*eppudu falaknuma express gunturuku veltundi* [When the Falaknuma Express goes to Guntur])

Here query is parsed based on spaces. After parsing each word, it is searched in the knowledge base until the word is found. After searching each word/term in the knowledge base, their types and semantic information are put in a list of tokens. Each token has three properties: the token value, its type and semantic information that it contains. These tokens and keywords are used to decide the proper query frame.

For the above example, the tokens identified are ఫలక్ నుమా ఎక్స్‌ప్రెస్ (Falaknuma Express) as Train name and గుంటూరు (Guntur) as Station name. Whereas ఎప్పుడు (e*ppudu* [when])**,** వెళ్తుంది (*veltundi* [goes]) are under keywords list.

3.4 Query Frame Decision

During the analysis of query, the keywords in the input query are detected. In this step, based on the tokens and keywords, we identify the appropriate *query frame.*

Restricting the query domain and information resource, the scope of the user request can be focused. That is, there are a finite number of expected question topics. Each expected question topic is defined under a single query frame. Some query frame examples for Railway information system are fare of a journey [Fare], arrival [Arr_Time] or departure time [Dep_Time] of a train, trains between important stations [Trains_Imp_Stations], scheduled time [Sched_Time], weekly frequency of a train [Arr_Frequency / Dep_Frequency], Availability of reservation class in a particular train [Reservation_Availability] and PNR enquiry [PNR_Enquiry].

It is important to select the appropriate query frame for the user request; because in some cases ambiguity will occur i.e. a single natural language query statement may belong to one or more query frames means same keywords are used to identify the query frames.

For example keywords like వెళ్ళు (*vellu* [*go*]), వచ్చు (*vachhu* [come]), చేరు (*cheru* [reach]), and బయలుదేరు (*bayuluderu* [start]) etc. are used to identify the query frames [Arr_Time], [Dep_Time], and [Trains_Imp_Stations]. To resolve this ambiguity, we consider what/which (question having words ఏ (*ee* [what]), ఏఏ (*eeee* [what]), ఏవి (*evi* [which]) etc.) type of questions like న్యూఢిల్లీ నుండి హౌరాకు బయలుదేరు రైళ్ళు ఏవి (*newdelhi nundi howrahku bayaluderu raillu evi* [What are the trains starts from New Delhi to Howrah]) are under [Trains_Imp_Stations] query frame. Where as, when (questions having words ఎప్పుడు (*eppudu* [when]), ఎన్నిగంటలకు (*ennigantalaku* [at what time]), ఎన్నింటికి (*ennintiki* [at what time]) etc.) type of questions like ఎన్నిగంటలకు కోల్కతా రాజధాని ఎక్స్‌ప్రెస్ బయలుదేరుతుంది (*ennigantalaku kolkata rajadhani express bayaluderutundi* [When Kolkata Rajdhani Express starts]) are under [Dep_Time] query frame. Similarly, weekday names like సోమవారము (*somavaaramu*) [Monday], మంగళవారము (*mangalavaaramu*) [Tuesday] etc. and keywords used in [Arr_Time]/ [Dep_Time] query frame are used to identify the [Arr_Frequency]/ [Dep_Frequency] query frame.

In contrast, separate keywords are used to identify [Arr_Time] and [Dep_Time] query frames. But keywords like పోతుంది (*potundi* [go]), వెళ్ళు (*vellu* [go]) etc. are used to identify both [Arr_Time] and [Dep_Time] query frames. To resolve this ambiguity, we consider the station type, i.e. whether the station is source or destination. If the station is source station (station name succeeded by postpositions like నుండి (*nundi* [from]), నుంచి (*nunchi* [from])), then we conclude that our query is under [Dep_Time] query frame. Otherwise query will be under [Arr_Time] query frame. For example, questions like ఎప్పుడు ఫలక్ నుమా ఎక్స్‌ప్రెస్ గుంటూరుకు వెళ్తుంది (e*ppudu falaknuma express gunturuku veltundi* [When the Falaknuma Express goes to Guntur]) is under [Arr_Time] query frame. But, questions like ఎప్పుడు ఫలక్ నుమా ఎక్స్‌ప్రెస్ గుంటూరు నుండి వెళ్తుంది (e*ppudu falaknuma express gunturu nundi veltundi* [When the Falaknuma Express goes from Guntur]) is under [Dep_Time] query frame.

The selection process of query frame has a great influence on the precision of the system, while there is not much likelihood of errors in other processes, such as getting the information from the dialogue history or generating SQL statement(s) from the selected query frame and/or retrieving the answer from the database and generating natural language answer from the retrieved result.

3.5 Dialogue Manager

The role of the Dialogue Manager (DM) differs slightly between different dialogue systems. But the primary responsibility of the DM is to control the flow of dialogue by deciding how the system should respond to a user request and the coordination of the other components in the system. If some information is missing or a request is ambiguous, clarification questions are specified by the DM and posed to the user.

For example in general, users ask questions about Arrival/Departure time without mentioning journey of train i.e. Upward/Downward journey, then system asks the user for proper information. Sometimes user may not give correct information (like missing Train name, Station name or query does not belong to any of the query frames etc.). At that time DM generates error message describing that missed information. In another case user asks questions without knowledge. In this case DM generates a *cooperative message,* which will help the user in further requests.

As a basis for the above tasks the DM utilizes the *dialogue history*. Here dialogue history records the focal information, i.e what has been talked in the past and what is talking at present. It is used for dialogue control and disambiguation of context dependent requests. The DM gets a semantic frame from the other system components. This frame is filled by interpreting the request in the context of the ongoing dialogue, domain knowledge, and dialogue history. The DM then prompts for missing information or sends a SQL query. Before the query is sent off, DM checks whether new information is contained in the query or the information is contradictory to information given before. If this is the case then the DM can either keep the original information or replace it with the new one in the dialogue history or engage in a confirmation subdialogue.

The DM looks at the query after language processing has been completed (but before the formal query is issued), as well as after the result has been obtained from the formal query. The accuracy of the system mainly depends on the representation of the dialogue history and how the DM responds to the user's dialogue.

3.6 SQL Generation

Once the query frame is selected for a question, the corresponding procedure for the SQL query generation is called. For each query frame there is a procedure for SQL statement(s) generation. In order to generate the SQL query, it needs the tokens generated by the query analyzer.

If the tokens are presented in the current query, it uses them. Otherwise it gets the token information from the dialogue history. For example, in the arrival time queries user has to specify Train name/no and station/city name where he/she needs to go. If he/she did not mention that information, SQL generation procedure gets the information from the dialogue history. Figure 2 depicts the conversion of natural language query to its SQL query.

ఎప్పుడు ఫలక్‌నుమా ఎక్స్‌ప్రెస్ గుంటూరుకు వెళ్తుంది (eppudu falaknuma express gunturuku veltundi When the Falaknuma Express goes to Guntur])?

↓

Train name: ఫలక్‌నుమా ఎక్స్‌ప్రెస్ (Falaknuma Express)
Station name: గుంటూరు (Guntur)
Keywords: ఎప్పుడు (e*ppudu* [when])**,** వెళ్తుంది (*veltundi* [goes]).

↓

The [Arr_Time] Query frame is selected.

↓

The system checks with the user for up/down journey of the train

↓

Let user asked about upward journey of train via DM.

↓

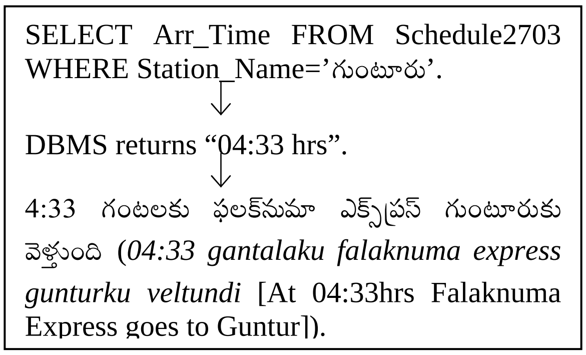
SELECT Arr_Time FROM Schedule2703 WHERE Station Name='గుంటూరు'.

Figure 2: Interpreting the natural language question to the SQL query

For the fare related query, SQL generation procedure would be called depending on the type of train. The procedure considers that the user will provide the train name and reservation class. If the train is of Express type, it considers that the user may provide either the source and destination stations of journey or the distance of journey. If it is of Rajdhani type, it considers that the user may provide source and destination station of journey. Similarly for the other query frames, SQL generation procedure considers that the user provide the necessary information.

3.7 Answer Generation

Once the SQL statement for an input query statement is generated, it is triggered on the database and the retrieved information is used to represent the answer. The retrieved information is updated in the dialogue history for further reference.



SELECT Arr_Time FROM Schedule2703 WHERE Station_Name='గుంటూరు'.

↓

DBMS returns "04:33 hrs".

↓

4:33 గంటలకు ఫలక్‌నుమా ఎక్స్‌ప్రస్ గుంటూరుకు వెళ్తుంది (*04:33 gantalaku falaknuma express gunturku veltundi* [At 04:33hrs Falaknuma Express goes to Guntur]).

Figure 3: Generating answer from the retrieved result.

Each query frame has its corresponding Answer generator. We use template based answer generation method. Each template consists of several slots. Those slots are filled by the retrieved answer and the tokens generated from the query. Figure 3 shows the answer generation from the SQL query generated from the natural language query shown in Figure 2. The answer will be sent to Dialogue Manager, which will further send it to the user.

4 Evaluation

For evaluating our system we have taken queries from our Telugu-speaking friends. We have described the Railway Information system to them. They have also been told about the constraints on the nature of queries in the systems. They have also been shown list of example queries for the systems.

Here we are considering two measures for evaluating our system: *Dialogue success rate* and *Precision.* The QA system was evaluated by giving 26 sets of dialogue consisting 95 natural language queries in total. The two evaluation measures are defined as follows:

Dialogue success rate for each set=Number of Answers or Responses generated by the system /Number of turns issued by the user.

Dialogue success rate = ($\sum$ Dialogue success rate for each set / Number of sets of dialogues)*100.

Precision= (Number of correct answers given by the system/Number of answers given by the system)*100.

The number of turns issued by the user in a dialogue is the total of the number of questions issued to the system and the number of responses provided by the user to the system.

Each set of dialogue consisted of around 3 to 5 natural language queries. The total dialogue success rate for the 26 sets was obtained as 21.83. The dialogue success rate for the system is calculated as

Dialogue success rate= (21.83/26)*100= 83.96%.

Out of 95 questions, system generated answers for 82 questions of which 79 were correct answers. So, the precision of the system is calculated as

Precision= (79/82)*100= 96.34%.

This low dialogue success rate is due to the fact that the system coverage of the domain is not extensive enough, i.e., query frames for some natural language queries were not correctly identified. The information given by the user in the query was sometimes inadequate and the system was not able to identify the missing information because of the incorrect choice of the query frame. Sometimes the system is unable to obtain tokens correctly from the input query even if it had identified the right query frame, thereby generating wrong answers. Misinterpretation of dialogue history is also another problem.

5 Conclusion

In this dialogue based QA system following the keyword based approach, each word need not be found in the knowledge base. Only the words that contain semantic information needs to be found in the knowledge base.

By restricting the coverage of questions, our system could achieve relatively high dialogue success rate. However, for a real practical system this success rate must be improved.

In extension to our work we are developing the modules for the remaining query frames. The system needs to be upgraded so that a user can query for railway information over phone. The speech input can be converted to textual query. This textual query can be input of our system and the textual out can be converted to speech again to answer the user.

References

Androutsopoulos I, Ritchie G. D, and Thanisch P. 1995. Natural Language Interfaces to Databases –

An Introduction. *Natural Language Engineering*, Vol 1, Part1, 29–81.

Diekema A.R, Yilmazel Ozgur, and Liddy E.D. 2004. Evaluation of Restricted Domain Question-Answering Systems. In *Proceedings of the ACL2004 Workshop on Question Answering in Restricted Domain,* 2-7.

Doan-Nguyen Hai and Leila Kosseim. 2004. The Problem of Precision in Restricted-Domain Question Answering. Some Proposed Methods of Improvement. In *Proceedings of the ACL 2004 Workshop on Question Answering in Restricted Domain*, 8-15.

Flycht-Eriksson Annika and Jonsson Arne. 2000 Dialogue and Domain Knowledge Management in Dialogue Systems. In *proceedings of 1st SIGDIAL workshop at ACL2000.*

Green W, Chomsky C, and Laugherty K. 1961. BASEBALL: An automatic question answerer. *Proceedings of the Western Joint Computer Conference*, 219-224.

Hoojung Chung, Young-In Song, Kyoung-Soo Han, Do-Sang Yoon, Joo-Young Lee, Hae-Chang Rim and Soo-Hong Kim. 2004. A Practical QA System in Restricted Domains. In *Proceedings of the ACL 2004 Workshop on Question Answering in Restricted Domain*, 39-45.

Hovy E, Hermjakob U, Lin C.Y, and Ravichnadran D. 2002. Using knowledge to facilitate factoid answer pinpointing *.COLING.*

Voorhees E.M. 2004. Overview of the TREC 2003 question answering track. In *Proceedings of the 12th Text REtrieval Conference.*

Woods W.A, Kaplan R.M, and Webber B.N.1972 The Lunar Sciences Natural Language Information System: Final Report. BBN Report 2378, *Bolt Beranek and Newman Inc.*, Cambridge, Massachusetts.

Xu J, Licuanan A and Weischedel R. 2003. TREC 2003 qa at bbn: Answering definitional questions. *TREC.*

Appendix: Examples of QA in the system

Example 1:
U: ఎన్నిగంటలకు కోల్‌కతా రాజధాని ఎక్స్‌ప్రెస్ బయలుదేరుతుంది (*ennigantalaku kolkata rajadhani express bayaluderutundi* [When Kolkata Rajdhani Express starts])?
S: పైకా(*pikaa* [is it up])/క్రిందికా (*krindikaa* [is it down])
U: పైకు (*piku* [up])
S: 16:15 గంటలకు కోల్‌కతా రాజధాని ఎక్స్‌ప్రెస్ హౌరా నుండి బయలుదేరుతుంది (*16:15 gantalaku kolkata rajadhani express howrah nundi bayaluderutundi* [At 16:15hrs Kolkata Rajdhani Express starts from Howrah]).

Example 2:
U: కోల్‌కతా రాజధాని ఎక్స్‌ప్రెస్‌లో ఫస్ట్ ఎసికు టుటైర్ ఎసికు హౌరా నుండి మొఘల్‌సరాయ్‌కి టిక్కెట్టు ధర ఎంత
(*Kolkata rajadhani expressloo first acku twotier acku howrah nundi mughulsaraike tikkettu dhara enta* [What is the fare between Howrah to Mughalsarai in Kolkata Rajdhani Express for First A.C, Two Tier A.C])?
S: కోల్‌కతా రాజధాని ఎక్స్‌ప్రెస్‌లో ఫస్ట్ ఎసికు టుటైర్ ఎసికు హౌరా నుండి మొఘల్‌సరాయ్‌కి టిక్కెట్టు ధర 2061, 1060 రూపాయిలు. (*Kolkata rajadhani expressloo first acku twotier acku howrah nundi mughulsaraike tikeettu dhara 2061, 1060 rupaayulu* [Fare from Howrah to Mughulsarai in Kolkata Rajdhani Express for First A.C, Two Tier A.C is 2061, 1060 respectively]).
U: స్లీపర్ క్లాస్‌కు (*sleeper classku* [to Sleeper Class] (Here it refers to the previous request of the user)
S: కోల్‌కతా రాజధాని ఎక్స్‌ప్రెస్‌లో ఫస్ట్ ఎసి, టుటైర్ ఎసి , త్రీటైర్ ఎసి మాత్రమే ఉంటాయి. (*kolkata rajadhani expressloo first ac, twotier ac, three tier ac matramee untayi* [Rajdhani Express has First AC, Two Tier AC and Three Tier AC only]). Because in India, Rajdhani Express won’t has Sleeper Class. This cooperative message generated by the DM.

Example 3:
U: సికింద్రాబాద్ నుండి తిరుపతికు ఏఏ రైళ్ళు ఉన్నాయి (*secundrabad nundi tirupatiki eeee raillu unnayi* [What are the trains from Secunderabad to Tirupati])?
S: కృష్ణా ఎక్స్‌ప్రెస్ (Krishna Express),శబరి ఎక్స్‌ప్రెస్ (Sabari Express), పద్మావతి ఎక్స్‌ప్రెస్ (Padmavati Express), నారాయణాద్రి ఎక్స్‌ప్రెస్ (Narayanadri Express).
U: ఏవి సోమవారము బయలుదేరుతాయి (*evi somavaaramu bayaluderutayi* [Which start on Monday])?
S: కృష్ణా ఎక్స్‌ప్రెస్ (Krishna Express),శబరి ఎక్స్‌ప్రెస్ (Sabari Express), నారాయణాద్రి ఎక్స్‌ప్రెస్ (Narayanadri Express).
U: ఎన్నింటికి నారాయణాద్రి ఎక్స్‌ప్రెస్ వెళ్తుంది (*ennintiki narayanadri express veltundi* [When Narayanadri Express goes])?

S: 07:15 గంటలకు నారాయణాద్రి ఎక్స్‌ప్రస్ తిరుపతి వెళ్తుంది (*07:15 gantalaku narayanadri express tirupati veltundi* [At 07:15 hrs Narayanadri Express goes to Tirupati]).

Example 4:
U: సికింద్రాబాద్ నుండి తిరుపతికు ఏ రైళ్ళు వెళ్తాయి (*secundrabad nundi tirupatiki ee raillu veltayi* [What are the trains from Secunderabad to Tirupati])?
S: కృష్ణా ఎక్స్‌ప్రస్ (Krishna Express),శబరి ఎక్స్‌ప్రస్ (Sabari Express), పద్మావతి ఎక్స్‌ప్రస్ (Padmavati Express), నారాయణాద్రి ఎక్స్‌ప్రస్ (Narayanadri Express).
U: ఏది త్వరగా వెళ్తుంది (*edi twaragaa veltundi* [Which Goes Early])?
S: No answer. Because keyword త్వరగా (*twaragaa* [early]) was not found in the knowledge base.i.e query belongs to another query frame.